UNIVERSITY OF OSLO
Department of Informatics

Why Not!

Sequence Labeling the Scope of Negation Using Dependency Features

Master's thesis

Emanuele Lapponi

August 1, 2012

Abstract

In this work we present a CRF-based sequence-labeling system for negation scope resolution that relies heavily on syntactic information extracted from dependency graphs. The models for negation are obtained using two corpora that differ from each other both in terms of language domain and scope annotation, allowing us to present parallel, comparative results for system configurations that draw information from similar, yet conceptually different sources. We evaluate the performance of the system on several levels. First, we assess the utility of syntactic features and label sets of different granularity, showing how the benefits of more involved configurations vary across corpora. Then, we compare our best performing configuration to similar systems, showing that our approach outperforms all known CRF-based systems on the same corpora. Finally, we evaluate the performance of system configurations based on different negation models as subcomponents of a simple engine for Sentiment Analysis.

Acknowledgements

First and foremost, I cannot possibly overstate my gratitude to **Lilja Øvrelid** and **Jonathon Read**, whose guidance and encouragement were invaluable for the completion of this thesis. I would also like to thank them, together with **Erik Velldal**, for believing in me enough to allow me to work with them on the 2012 *SEM shared task.

I wholeheartedly thank **Johan Benum Evensberget** and **Arne Skjærholt**, for taking the time to help me when the trees seemed too tall to climb and the probabilities just wouldn't sum up to one.

I am also grateful to my other lab companions, for the fruitful discussions and healthy breaks that positively affected my research: **Ron Unhammer**, **Jan Ole Skotterud**, **Lars-Erik Bruce**, **Murhaf Fares**, **Rune Lain Knudsen**, **Sindre Wetjen**, **Lars Jørgen Solberg**, **Runar Furenes**, **Charlotte Løvdahl**, **Kyrre Havik Eriksen**, **Tobias Vidarssønn Langhoff**, **Trond Thorbjørnsen** and **Joakim Boarding**.

My gratitude goes also to **the LT Group** as a whole, for fostering a great, inclusive learning environment for I:SK master students.

I am very grateful to **Isaac Councill**, **Ryan McDonald** and **Leonid Velikovich** for making the Product Review corpus available to me, and to **Roser Morante** and **Eduardo Blanco** for organizing the 2012 *SEM shared task, which was a major turning point for this work.

I would also like to thank the developers of Maltparser, Wapiti and NLTK.

Last, but certainly not least, my gratitude goes to **Ina**, for reasons that are impossible to synthesize here.

In memory of Mauro Salomone, computational astrophysicist, sword master, and beloved uncle.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the introduction of the book "The expression of negation" (Horn 2010), Laurence Horn writes:

> "Negation is a sine qua non of every human language but is absent from otherwise complex systems of animal communication. While animal *languages* are essentially analog systems, it is the digital nature of natural language negation, toggling between 1 and 0 (or T and F) and applying recursively to its own output, that allows for the essential properties of our own linguistic systems. In many ways, negation is what makes us human, imbuing us with the capacity to deny, to contradict, to misrepresent, to lie, and to convey irony."

Beside being a species-defining phenomenon, negation is strongly expressive: in its most obvious instance, it reverses the truth value of a statement (Example 1.1), while in more subtle examples it is a fierce conveyor of euphemisms and irony (Example 1.2, where (a) and (b) have the same meaning). In turn, this expressivity of litotes and euphemisms is reflected in so called *neg-raising* (Example 1.3), the "[...] subordinate clause understanding of certain main-clause negatives" (Horn and Yasuhiko 2000) and the use of multiple negative constructions (Example 1.4).

(1.1) (a) He was guilty

(b) He was not guilty

(1.2) (a) Very good!

(b) Not bad!

(1.3) (a) I don't think she'll come tonight

(b) I think she won't come tonight

(1.4) (a) Not unlike Microsoft

(b) just like Microsoft

Negation has a scope: morphologic negation (morphemes that invert the truth value of a word, like the affix *un-* in *unpleasant*) often negates only the rest of the word it is contained within, whereas syntactic negation might propagate to more than the first word on the right of the negation cue, the *operator* of negation, like the adverb *not*.

(1.5) Mario is not tall but he is fat.

(1.6) Mario is not tall and fat.

(1.7) Not only is Mario fat, he is also short.

(1.8) Mario is not just fat, he is short.

(1.9) Mario does not kill Bowser because they are friends.

In Example (1.5), one might argue that the scope of the negation cue *not* is limited to *tall* while (1.6) is ambiguous: *not* can either negate the whole verb phrase *is not tall and fat* or only *tall*. The ambiguity gets even more interesting in (1.9), where Bowser's life depends on the exclusion of the the adverbial modifier *because they are friends* from the negation scope. In (1.7) and (1.8), arguably nothing is negated but *only* and *just*. In Tottie's comprehensive taxonomy of English clausal negation (Tottie 1991) each category presents differences in the intended scope: the most interesting divergence is probably between *denials* (examples of which have been shown above) and *rejections* such as (1.10), where the answer negates the content of the question preceding it.

(1.10) Is the princess in this castle? No.

What should be considered to be the scope of negation? With all these examples in mind, this seems like a warranted question. Should the effects of a negation cue be assumed to propagate to a whole proposition, or is it a better idea to consider only the part of the proposition that is actually negated? Consider the following example:

(1.11) I was not awake in class.

It is perfectly reasonable to say that only *awake* has its truth value reversed by *not*; we could paraphrase the sentence with its semantic opposite *sleeping* and produce the equivalent "I was sleeping in class". It is however just as reasonable to say that the whole proposition is negated, an interpretation that becomes apparent when we paraphrase it with "It was not the case that *I was awake in class*".[1]

Certainly such multifacetedness, intricacy and expressivity makes the study of negation very interesting from a purely linguistic point of view. It also makes it a very challenging and important problem in the context of Natural Language Processing (NLP), a multi-disciplinary research field that focuses on the computational treatment of human language. NLP applications permeate our digital lives, enabling search engines to provide increasingly relevant results and recommendation systems to know what we "might be interested in". Speech recognition, syntactic and semantic parsing allow systems like Apple's *Siri* to accept queries in plain English and react accordingly.

All of these systems can undoubtedly benefit from knowing what parts of the information they process are affected by negation.

1.1 Thesis

The aim of this work is to contribute to the ongoing research on negation in the Language Technology community. We present a machine learning-based system that applies sequence labeling techniques to negation scope resolution, relying heavily on syntactic information obtained from dependency graphs.

Whereas this is not a novel approach, our contribution lies in investigating the performance of this system on several levels, paying special attention to the role played by syntactic features and the granularity of label-sets. Perhaps most interestingly, we utilize two different corpora for training and testing our system; these differ from each other not only in terms of domain, but also in terms of the notion of negation scope itself. This allows us to present parallel, comparative results for system configurations that draw information from similar, yet conceptually different sources.

The performance of configurations built on these different ideas of scope are finally put to the test as subcomponents in a simple system for Sentiment Analysis, wherein the opinion represented by text is classified as broadly positive or negative.

[1] After all, that is how "not" jokes work: "I was awake in class — *Not!*"

Audience

While the primary intended audience of this work are other NLP researchers, we have attempted to structure this thesis so that it can be accessible to any computer science student with a healthy interest in human language. In this respect, we hope that this work may serve as a concrete example of how a natural language processing system can be conceived, engineered and evaluated.

Chapter overview

Chapter 2 serves as an introduction to machine-learning techniques for sequence labeling. We present one of the simplest generative models, the Hidden Markov Model, both to provide an intuitive introduction to these techniques and to motivate the adoption of Conditional Random Fields as the core component of our system.

Chapter 3 is dedicated to grammar. Much of the feature engineering presented in this thesis is based on dependency parses, and part of the experiments conducted are designed to investigate the effects of dependency features on the sequence labeling model. After briefly motivating the usefulness of grammatical formalisms, this chapter provides an introduction to phrase and dependency structures, and explains how dependency graphs are automatically obtained with a deterministic shift-reduce algorithm.

Chapter 4 presents related work that inspired and motivated this thesis and the corpora used to build the system that accompanies it. Here we provide important, comparative insights and details on the nature of these two corpora.

Chapter 5 is an in-depth description of our negation scope resolution system, explaining every step of the data processing in order to facilitate result replication. Here we motivate our choices in terms of both feature engineering and choice of labels, present a strategy for pseudo-gold negation-cue extraction in one of our corpora and explain how multiple layers of information are collapsed (and expanded) into simple sequences of labels.

Chapter 6 explains how our system is evaluated and clarifies some points concerning the experimental setup.

Chapter 7 details the experiments performed, presenting the results obtained and our reflections around them.

Chapter 8 is an excursion in Sentiment Analysis, another area of research within the NLP sphere; to extrinsically evaluate the potential of different configurations of our system, we build a simple Sentiment Analysis engine where our system serves as a sub-component.

Chapter 9 sums up the outcomes of the work done in this thesis and considers the possibilities for future work.

Chapter 2

Sequence Labeling

Negation scopes can be seen as *chunks* of text of varying length. Examining the sentence in Example 2.1 without concentrating on its grammatical subtleties or semantic qualities, focusing instead on its appearance based on the parameters of negation alone, reveals the following: some words are in scope (labeled with the letter N) and others are not (labeled with O).[1] Negation cues (labeled with CUE in the example) can be seen as special words as well, since they signal scopes.

(2.1)

I	'm	afraid	I	ca	n't	do	that	,	Dave	.
O	O	O	O	O	CUE	N	N	O	O	O

Automatic text chunking has a relatively long tradition within the NLP realm (Jurafsky and Martin 2000), and IOB-chunking (for Inside, Outside and Begin) is a standard technique used to obtain shallow parses. Rather than using full-fledged syntactic parsing, a supervised machine learning model can be used to label sequences of text and labeling phrases like the NP *my good friend Mario* in Example 2.2.

(2.2)

My	good	friend	Mario	jumps	very	often	.
B	I	I	I	O	O	O	O

The determiner *My* is labeled with a B because of its initial position in the phrase; multi-word NPs tend to start with determiners like the possessive *my*, the determinative article *the* or the demonstrative *that*. As a whole, NP sequences possess certain characteristics, or *features*, that set them apart from other chunks like Verb Phrases, which usually start with verbs. The kind of label assigned to each token in the sentence can be seen as dependent on its surrounding context,

[1] Note that Example 2.1 is labeled according to the specifications of the Product Review Corpus, which are detailed in Chapter 4.

both in terms of words, other labels and additional hidden layers of information like parts of speech and syntactic constituents. This dependence can be modeled using probabilities, which can be learned from a corpus of annotated sequences and used to automatically annotate new data.

In this work, negation scope resolution is treated as a special instance of the IOB sequence labeling problem; since some familiarity with these techniques is essential to understand how the system presented in Chapter 5 works, the rest of this chapter will provide a brief introduction to supervised machine-learning techniques for sequence labeling.

2.1 Hidden Markov Models

One of the simplest machine-learning techniques for sequence labeling is the Hidden Markov Model (HMM), which enables the computation of the most probable label sequence $\hat{t}_1^n$ given a sequence of words w_1^n, i.e.:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

Where $\operatorname{argmax}_{t_1^n}$ can be read as "the t_1^n that gives the highest probability". The conditional probability $P(t_1^n|w_1^n)$ can be calculated by refactoring the right side of the equation with a set of other probabilities using Bayes theorem:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)} = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n) \tag{2.3}$$

Based on observations from a training corpus, $P(w_1^n|t_1^n)$ is the *likelihood* that a sequence of words w_1^n is *emitted* by a sequence of labels t_1^n, while $P(t_1^n)$ is the *prior probability* that the sequence of labels will occur in the first place. The denominator $P(w_1^n)$ can be dropped since the sequence of words is the same for all the possible sequences of labels.

As it is, Equation 2.3 is too hard to compute. In HMMs, two independence assumptions are made. First, the single emissions in the sequence are assumed to be conditionally independent from each other. By virtue of the chain rule, the probability of the whole sequence is approximated to the product of the probabilities of the single emissions:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i) \tag{2.4}$$

Secondly, the probability of a tag is assumed to depend only on the previous *n* tags, where *n* is a number sufficiently small to keep the computation tractable. With $n = 1$, we have a bigram HMM:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}) \tag{2.5}$$

Substituting equations 2.4 and 2.5 into 2.3 we get the following expression for the most probable tag sequence:

$$\hat{t}_1^n \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}) \tag{2.6}$$

The probabilities $P(w_i|t_i)$ and $P(t_i|t_{i-1}$ in the left sides of equations 2.4 and 2.5 are Maximum Likelihood Estimates (MLE) that are calculated using the training corpus. Calculating MLEs is the *learning* phase in an HMM-based system. The probability of transitioning from a label to another is calculated by counting how many times t_{i-1} is observed to be immediately preceding t_i out of the times it is observed on its own:

$$P(t_i|t_{i-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})} \tag{2.7}$$

Similarly, the probability of a label t_i emitting a word w_i is simply the number of times w_i is labeled with t_i out of the times w_i appears in the corpus:

$$P(w_i|t_i) = \frac{c(t_i, w_i)}{c(w_i)} \tag{2.8}$$

HMMs are widely adopted in the context of PoS tagging. Consider the following example: with a sufficiently large training corpus, the MLE in 2.7 assigns a higher probability to a noun like *cat* if the previous tag is a determiner like *the*. On its own, assuming that determiners are most commonly followed by nouns in the training corpus, the transition probability would consistently assign a noun tag if the previous one is a determiner, so *the good cat* would end up being mislabeled. The emission probability in 2.8 would amend to this; while adjectives have lower probability of following determiners than nouns, *good* has a much higher probability of being an adjective than a noun[2] — combining the transition and emission probabilities, the correct sequence is retrieved.

Applying an HMM to learn the CUE, N and O labels in Example 2.1 might not be an equally good idea. Consider the unbalance in the frequency distribution of the labels in the Product Review corpus, where 95 percent of the tokens are

[2]Though it *can* be a noun, as in "I am doing this for your own good". In general, whether it has been observed in the training set or not, no transition or emission is considered to be impossible, just very unlikely.

out of scope. While transitioning from a CUE to an N would surely be more probable than to an O, the rest of the transition probabilities would favor Os, since the vast majority of words in the training corpus is out of scope. Emission probabilities would provide little help to adjust this one-sidedness; while it is reasonable to assume that a CUE is more likely to emit *not* or *nothing* rather than *cat* or *good*, we can expect Os to be more likely than N to emit any word.

Fortunately, there is more information to be gathered in the context of scope resolution; we could, for instance, assign an integer to each token in the training corpus to represent the distance d_i from a negation cue, behind the intuition that the probability for a word to be affected by a negation cue decreases as the distance between the two increases. Calculating the MLE of increasingly complicated patterns, however, would yield progressively sparser distributions.

In order to properly address scope resolution we need another strategy, one that can take advantage of more detailed features in a meaningful way.

2.2 Conditional Random Fields

The HMM that we just presented is a *generative* model, one that describes how output labels are probabilistically generated as a function of an input observation. A *discriminative* model allows the direct modeling of the conditional distribution $P(\mathbf{y}|\mathbf{x})$, where $\mathbf{y}$ is some output we want to find and $\mathbf{x}$ some collection of arbitrarily many observations. Conditional Random Fields (CRFs) belong to this family of models, and they are most commonly used in their linear-chain form for sequence labeling tasks.

Sutton and McCallum (2010) define CRFs as follows:

> Let Y, X be random vectors, $\theta = \{\theta_k\} \in \Re^K$ be a parameter vector, and $\{f_k(y_t, y_{t-1}, \mathbf{x}_t)\}_{k=1}^K$ be a set of real valued feature functions. Then a *linear-chain conditional random field* is a distribution $p(\mathbf{y}|\mathbf{x})$ that takes the form
>
> $$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\} \tag{2.9}$$
>
> where $Z(\mathbf{x})$ is an instance-specific normalization function
>
> $$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\} \tag{2.10}$$

In simpler terms, this means that at each point of a T-long sequence we consider K-many feature functions that can be either active or inactive. For

instance, one such feature function could mirror the question "is the current word in the sequence a noun with distance 1 from a negation cue?". If the feature is active (answering "yes, it is!" to our question) it returns 1, 0 otherwise. The value returned by the feature function is multiplied with a weighting parameter θ_k that is associated to it and has been learned during training.

These weights guide the labeler in deciding which label is most probable. The theory behind parameter estimation in CRFs is however more involved than calculating MLEs for Hidden Markov Models; should the reader be inclined to dive in the specifics, we recommend the introductions by Sutton and McCallum (2010) and Skjærholt (2011).

The negation scope resolution system developed in conjunction with this thesis utilizes a CRF labeler, allowing us to consider elaborate lexical, syntactic and contextual features like the ones extracted taking advantage of the dependency graphs presented in the next chapter.

Chapter 3

Dependency Grammar

Much like their grownup counterparts in the natural sciences department, most children can't utter the word *grammar* without rolling their eyes in contempt. The first skill they acquire from their grammar classes is that of assigning words with their associated semantics to a class: things you can *do* are verbs, things you can *touch* and *feel* are nouns, words that describe nouns are adjectives and those that describe verbs are adverbs. Then, they are told that the classes are more fine-grained: ask an italian ten year old to classify "fossi stato" and chances are she will answer *verb:first-person-singular-pluperfect-conjunctive*.[1] That was only morphology: by the time the younglings start to learn about syntax, or how the order and the form of these words depend on those of the others, some free spirits have started to protest — "Why do we have to know this stuff? We can speak and write perfectly well already." — they say. There is a number of answers to this question, a perfectly valid one being the following: by learning formalisms that describe the way humans express intricate thoughts with language, we can gain a better understanding of the way the human mind works. It might very well be that these formalisms will help us answer an existential question or two.

On the pragmatic side, and arguably more likely to excite certain children with a penchant for science fiction, grammar is an essential tool for Natural Language Processing tasks that attempt to mimic a glimpse of human cognition. Ambiguity is a classic example. Morphology alone can sometimes help us disambiguate $face_{verb}$ and $face_{noun}$ without even looking at the surrounding context. Where morphology isn't enough, syntactic annotations provide a mean to computationally represent and make sense of ambiguous propositions such as "I shot an elephant in my pajamas".[2] Syntax gives us essential tools to write programs that are not entirely helpless when it comes to understanding *who does what to whom*, because it allows us to learn the kind of patterns we can expect to find in

[1] The day before the final primary-school exam is an especially good day to ask.

[2] Who is wearing the pajamas, me or the elephant?

the infinite set of grammatical sentences in a given language. It is therefore very practical that both linguists and computer scientists use the same data structures to represent interconnected data.

In the following, we will introduce phrase and dependency structures. Since features extracted by taking advantage of dependency graphs, we will devote the last section of this chapter to explain how automatic dependency parsing works.

3.1 Phrase structures

One very influential grammar formalism, baptized *Phrase Structure Grammar* by Chomsky (1957), is based around a set of rewriting rules that construct a syntactic tree, a data structure originating from terminal nodes, the *leaves*, that culminates in a whole grammatical sentence, represented by the *root* node. The grammar in Table 3.1 allows symbols on the right side of the arrows to be combined into the ones on the left side. A combination of symbols is a *phrase* (also known as *constituent*); phrase labels can be found on either side of the arrow, making the grammar recursive.

$$\begin{aligned}
&\text{NP} \rightarrow \text{PRP} \\
&\text{NP} \rightarrow \text{DT NN} \\
&\text{VP} \rightarrow \text{VBD NP} \\
&\text{S} \rightarrow \text{NP VP PU}
\end{aligned}$$

Table 3.1: Example grammar

Consider the sentence “I needed no friends”. In order to analyze its syntax, we first need to acquire each part of speech, a first level of abstraction that gives us symbols to combine according to the rules in Table 3.1; their recursive application can be represented by a syntactic tree like the one in Figure 3.1.

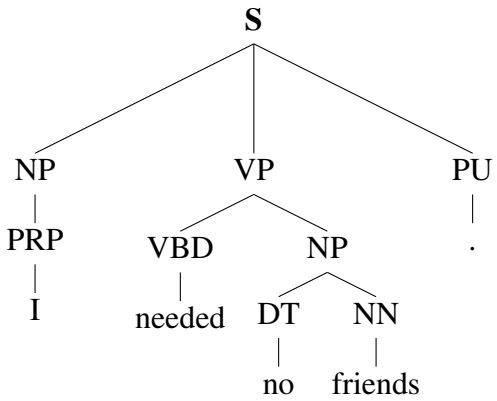

Figure 3.1: Example Syntactic Tree.

3.2 Dependency structures

Dependency grammar is a representation that differs from phrase structures in several aspects. Tesnière (1959) provides a description of the idea behind dependency grammars (Nivre 2005):

> "The sentence is an organized whole, the constituent elements of which are words. Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence. The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. The superior term receives the name governor. The inferior term receives the name subordinate. Thus, in the sentence *Alfred parle* [...], *parle* is the governor and *Alfred* the subordinate."

This means that this grammar revolves around the concept of syntactic *heads*, and each syntactic structure is represented by binary, directed relations between words. These relations link heads to their dependents in order to form constructions, where each dependent is allowed to have exactly one head. there are several syntactic and semantic criteria we can use to determine head status, many of which are described in Zwicky (1985).

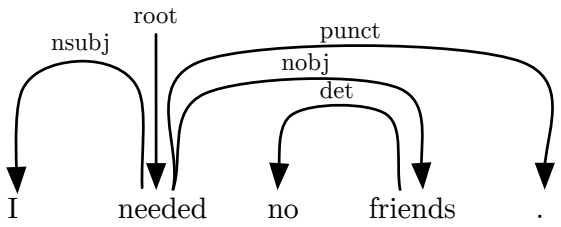

Figure 3.2: Example dependency graph.

To analyze the sentence in Figure 3.2 we can rely on *valency binding* to decide that *needed* is the head of both *I* and *friends* because they are its licensed subject and object; the verb decides their positions and their semantic category, because the predicate *need* requires both a *needer* and what it *needs*. To determine the direction of the dependency relation between *no* and *friends* we can rely on one of the criteria postulated by Zwicky, which says that a head determines the syntactic category of the whole construction and can replace it. In this

case, *no* can be ablated resulting in a sentence that is still grammatical, while the same is not true for *friends*. There is a number of such criteria and some finer-grained distinctions between construction types; a complete overview is however not necessary for the purpose of this work.[3]

Assuming the formalization of Nivre (2005), once we have connected all the tokens in the sentence, the resulting graph is acyclic and directed, meaning that all the nodes in the graph are hierarchically ordered by the edges and their direction and there is no path that crosses the same node more than once. A dependency graph can then be formally described as a graph $G = (V, E, L)$ where:

1. $L \subseteq R$
2. $V = \{v_{root}, v_1, ..., v_n\}$
3. $E \subseteq V \times V$
4. $\forall_x(e_x) \rightarrow \exists(l \in L)$

The first statement means that the set L of *labels* is a subset of all the dependency relations. The second describes the set V of nodes that contains n elements, where n is the number of words in a sentence, plus a *root* node. The third statement says that the set E of directed edges is a subset of the powerset $V \times V$, which means that it contains the arrows e_{xy} that link v_x to v_y. The last sentence simply states that for each edge there is a label l in L. The graph is *well formed iff* the edges in E create a fully connected graph that has, as already mentioned, a hierarchical order.

Dependency representations are more compact than phrase structures, since the number of nodes is constrained on the number of tokens in the sentence: there are 14 nodes in Figure 3.1, compared to 5 nodes in in Figure 3.2. This is rather advantageous from a programmer's perspective: while the asymptotic complexity of traversing both data structures is essentially the same, developing programs that take advantage of the provided information within a more constrained representation is, in some respects, a more straightforward affair. On the other hand, the deeper structural description of the sentence provided by the nested constituents is lost in dependency graphs. Still, the labels of each dependency arc represent functional categories that are closer, when not equivalent, to the semantics of the parts of a sentence: in Figure 3.2 there are two arrows that explicitly say *who* needs and *what* is needed, while in 3.1 this information has to be derived by processing the syntactic tree.

[3]For instance, constructions where the head can replace the whole are called *endocentric*. In *exocentric* constructions this is not the case, an example being the relation between *without* and *keeping* in Figure 5.4.

3.3 Dependency parsing

Dependency graphs can be obtained automatically using a parser, a program that takes a sentence as input and returns an annotated graph. The output of a Maltparser (Nivre et al. 2007), which is the primary source of syntactic information used in this work, outputs dependency graphs in the format pictured in Figure 3.3.

id	token	head	dep. relation
1	I	2	nsubj
2	needed	0	null
3	no	4	det
4	friends	2	nobj

Figure 3.3: An example of Maltparser's output, where each token is annotated with the id of its head and label of the arc that links it to it.

Maltparser is a dependency parser that can be described with the following attributes: data-driven, deterministic and transition-based. The rest of this section will explain how it works and what each of these attributes mean.

A traditional parser is built on a set of grammatical rules like the ones in Table 3.1; one that is data driven relies on a model induced with the probabilities extracted from a *treebank*, a large collection of text that has been manually annotated with syntax. Analyses that are more or less probable rather than right or wrong result in increased robustness, i.e. the parser is always capable to produce a dependency graph that satisfies the formal requirements described in Section 3.2. This robustness is essential to real-life parsing applications where a lot of unedited, user generated content needs to be processed, and is one reason why statistical methods are used in most state-of.the-art syntactic parsers.

The parsing algorithm in Maltparser is *transition-based*, meaning that words are processed through a series of transitions. Maltparser uses an adapted version of the *Shift-Reduce* algorithm, where the sentence is processed from left to right using a stack S of partially processed tokens and a queue Q of remaining ones, assigning arcs by taking one of the four possible actions in Figure 3.4.

Shift moves tokens from the start of Q to the top of S, while *Reduce* pops w_i from the top of the stack provided that it has been assigned a head. *Left-arc* makes the word w_i on top of S a dependent of the first word w_j in Q and immediately pops the stack, while *Right-arc* makes the top of S the head of the first token w_j in Q and shifts w_j to the top of S. The algorithm thus described is *non-deterministic*, since more than one of these actions is possible for a given

$$\text{Shift} \quad \frac{[\ldots]_S \quad [w_i, \ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q}$$

$$\text{Reduce} \quad \frac{[\ldots, w_i]_S \quad [\ldots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [\ldots]_Q}$$

$$\text{Left-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [w_j, \ldots]_Q \quad w_i \overset{r}{\leftarrow} w_j}$$

$$\text{Right-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_j}{[\ldots, w_i, w_j]_S \quad [\ldots]_Q \quad w_i \overset{r}{\rightarrow} w_j}$$

Figure 3.4: Shift-reduce parser actions showing the states of the stack and cue before and after each action.

configuration of S and Q. At non-deterministic choice-points, a classifier trained on the data from the treebank maps features of the current configuration and the partial graph to one of the parser actions in Figure 3.4. Popping the stack is the *greedy* step of the algorithm, which means that the local optimum has been found and no further processing of the node is necessary. The algorithm starts with an empty stack and a queue containing the whole sentence, terminating when all tokens have been popped from the stack and added to the dependency graph. The worst-case complexity of the algorithm is $O(n^2)$, where n is the number of tokens in the sentence.

Another common approach to data-driven parsing is to use exhaustive-search techniques like dynamic programming, resulting in slower processing compared to the algorithm described in this section. Much like robustness, speed is a very valuable property in real-life natural language processing, e.g. in machine translation, parsing of live speech or, in general, large amounts of web data.

3.4 Stanford dependencies and negation

The pre-trained model for English used in this work outputs Stanford typed dependencies (de Marneffe and Manning 2008b) in their basic format, where every dependency structure is directed and acyclic, forming a tree. This set of dependencies was designed with ease of use in mind, in order to allow researchers and professionals without an NLP background to extract meaningful textual relations.

In this spirit, negation is explicitly accounted for in this representation, albeit in a very basic form. The dependency relation *neg* links loosely defined "negation modifiers" (which, as we will see in the next chapter, correspond to adverbial modifiers labeled as negation cues in our corpora) to the word they modify — for instance in "Bill is not a scientist" there would be a *neg*-labeled edge going from *scientist* to *not* (de Marneffe and Manning 2008a).

Chapter 4

Corpora & Related Work

The body of work about negation in the field of linguistics is substantial: one might therefore be surprised to see that up until 2009 negation was a somewhat neglected area in Natural Language Processing, where most research was performed in the biomedical domain with a focus on whether a medical term is negated or not (Morante and Daelemans 2009). The last two years have seen a surge of interest in the matter, although until recently most efforts still revolved around the biomedical domain. This was probably due to the availability of the Bioscope Corpus (Vincze et al. 2008), a collection of biomedical reports, papers and abstracts annotated with negation and speculation cues with their scope.

The significance of work in this area was recently highlighted by the special issue of the Computational Linguistics journal, which features two articles that focus on negation. "Modality and Negation: An Introduction to the special issue" (Morante and Sporleder 2012) presents an overview of how negation has been treated from a computational perspective. "Speculation and Negation: Rules, Rankers, and the Role of Syntax" (Velldal et al. 2012) details the hybrid data-driven/rule-based approach that yields the best results to date in speculation resolution, and how it was ported to yield state-of-the-art results on negation as well.

Another very recent, notable endeavor that contributed to raise attention and interest to the topic of negation within the NLP community was the shared task organized in conjunction with *SEM, the first joint conference on lexical and computational semantics (Morante and Blanco 2012). This task was dedicated to resolving the scope and the focus of negation, and enabled researchers from several universities to directly compare the results of their systems. Participants could submit the outputs of their system to a closed track, where only annotations provided in the accompanying data-sets could be used, and to an open track, where additional linguistic information and tools were allowed. System outputs from 9 different teams — including our own — were submitted to the scope-

resolution subtask, presenting different approaches and techniques to tackle it.

For the work described in this dissertation, two systems predating the shared task, Morante and Daelemans (2009) and Councill, McDonald, and Velikovich (2010), provided inspiration; both attack the problem using supervised machine learning-based systems that treat the problem of negation resolution using sequence labeling techniques. In the rest of this chapter we will present an overview of these two systems before presenting the corpora used to develop our own.

4.1 Related work

In the Bioscope Corpus (BS), the annotations designate the boundaries of scopes and negation cues, as exemplified in 4.1.

(4.1)
```
<xcope id="X3.6.1"><cue type="negation"
ref="X3.6.1">Neither</cue> deletion of NF kappa B <cue
type="negation" ref="X3.6.1">nor</cue> deletion of NFAT-1
decreased activation of viral replication by phorbol
ester</xcope>.
```

The example shows a multi-word cue, *neither-nor*, and its scope, which spans through the whole sentence. The annotated scopes in Bioscope are *maximal*, meaning that they encompass the largest syntactical unit possible: for instance, in a sentence like "He does not eat meat.", the scope of *not* would be *He does eat meat* rather than only *eat meat*.

Morante and Daelemans (2009) use BS to train a machine learning-based system for negation scope resolution. Cues are classified relying on a closed class of words that function unambiguously as cues in the training corpus, complementing it with a cue classifier that considers the lemma and PoS of each word, the chunk of the two surrounding words and the word form of the second token in both directions. After classifying cues, the system resolves scopes by combining the output of three separate classifiers and feeding it to a fourth one. This rather involved classifier setup is presented with cue-token pairs and a plethora of lexical and syntactical features. Syntactic analysis is obtained via *chunking*,[1] and each token is represented with information that describes its relative lexical and syntactic position to the rest of the sentence and to the negation cue. The tokens thus represented are assigned to one of three classes: beginning of scope, end of scope or neither. After labeling, the output of the final classifier is post-processed to reconstruct the full scopes.

Councill, McDonald, and Velikovich (2010) develop a negation resolution system using BS as well, and use a sequence classifier to assign one of two

[1] As mentioned in Chapter 2, chunking is a form of shallow parsing that identifies constituents (noun groups, verbs, verb groups, etc.), but not their internal or hierarchical structure.

labels to each token in a sentence: *in* or *out* of scope. Cue classification is done through lookups in a lexicon of negation cues and each word in each sentence is, as in Morante and Daelemans (2009), represented by lexical and syntactic features. The latter are extracted via a dependency parser and are used both to describe the syntactic position of a token and to compute its distance from a negation cue in terms of the number of arcs that have to be traversed to get from one to the other. Table 4.1 contains the list of the cues in the lexicon, which was compiled analyzing word co-occurrence with n-grams like *either* or *at all*, so called *negative polarity items*, which signal the presence of negation.

hardly	lack	lacking	lacks
neither	nor	never	no
nobody	none	nothing	nowhere
not	n't	aint	cant
cannot	darent	dont	doesnt
didnt	hadnt	hasnt	havnt
havent	isnt	mightnt	mustnt
neednt	oughtnt	shant	shouldnt
wasnt	wouldnt	without	

Table 4.1: The list of negation cues used by Councill, McDonald, and Velikovich (2010) to detect cues in their system. It is compiled analyzing word co-occurrence with n-grams like *either* or *at all*, which signal the presence of negation.

This list is augmented with common misspellings (e.g. *aint*) and selected cues tagged with *Neg* or *Negate* in the General Inquirer (Stone 1966).

Both systems use Conditional Random Fields for assigning labels to the tokens[2]. Where Morante and Daelemans extract special instances of tokens that represent scope-start and scope-end, Councill et al. directly model the task as a sequence labeling one, where the task is to find the most probable label sequence $\mathbf{x}$ given a sequence of observations $\mathbf{y}$: the system developed for our research extends on their approach with regard to both classification scheme and use of dependency features.

[2]In Morante and Daelemans (2009) two of the four classifiers used for scope resolution are CRFs, including the meta-learner that takes the output of the other three as input.

4.2 Product Review

In addition to using the Bioscope corpus for development, Councill, McDonald, and Velikovich (2010) train and test their system on the Product Review Corpus (PR). This data set differs from BS with regard to annotation scheme, concept of scope and domain. Negation cues are not annotated; a frequency distribution of matches from the lexicon in Table 4.1 is presented Table 4.2, which shows that *not* and its contracted from *n't* account for 72% of matches.

n't	426	nobody	10
not	221	lack	9
no	75	none	8
never	44	didnt	3
nothing	28	dont	3
without	26	hardly	3
nor	16	doesnt	1
neither	12	isnt	1
nowhere	11	lacks	1

Table 4.2: Frequency distribution tokens in PR matching those in the lexicon of explicit negation cues in Table 4.1. Note that these are not equivalent with gold cues, which are not annotated in PR.

Unlike BS, scope annotations are minimal and are, in the vast majority of cases, to the right of a cue; in "He does not eat meat.", *eat meat* would be annotated as the scope of *not* rather than *He does eat meat*. A similar instance from the corpus is Example 4.2, where only the part of the sentence to the right of the negation cue is negated, while one might argue that the *ability* (i.e. *could*) to look at hip-hop the same is also affected by the cue.

This minimalism is apparent in noun phrases where only an adjective is negated, as in Example 4.3, or in cases where adverbs like *only* is the whole scope like in 4.5. Adverbial comparatives get a different treatment, as in Example 4.4, where the whole phrase is negated. From here on, examples will have bolded **cues** and underlined scopes; the following are taken directly from PR.

(4.2) [...] I could**n't** look at hip-hop the same.

(4.3) **Not** top-drawer Wodehouse, but still quite amusing and a good example of Wodehouse's "musical comedy without the songs.

(4.4) I gave it a poor review, mostly because it's **not** really my type of book, and sometimes I found the flowery language annoying.

(4.5) Had she chosen to reverse the acting choice, **not** only would Barbra would have given a more truthful and less conscious performance but it would have brought more credibility to a story already plagued with unbelievability.

Reviews	268
Sentences	2111
Sentences w/ negation	679
Scopes	730
Tokens	17,037

Table 4.3: PR Corpus statistics.

The corpus is comprised of 268 unedited reviews from Google Product Search, for a total of 2111 sentences, 679 of which contain negation; it was developed to model negation within the open-web realm. The language domain of PR makes it more apt to everyday-language processing than the biomedical reports and papers of BS, but comes with challenges of its own. For instance, mistakes like the repeated *would* in Example 4.5 are relatively common, and can effect the performance of taggers and parsers negatively. A sentence from an unprocessed review from PR is available in Figure 4.1, while Table 4.3 contains some corpus statistics. As Figure 4.1 shows, the annotations for sentences and negation scopes are provided in the form of xml-tags.

```
<sentence>Had she chosen to reverse the acting choice, not
<negation_span>only</negation_span> would Barbra would have given a more
truthful and less conscious performance but it would have brought more
credibility to a story already plagued with unbelievability.</sentence>
```

Figure 4.1: Review excerpt from PR, showing the xml-style annotations for sentence and scope boundary,

4.3 Conan Doyle

The data sets released in conjunction with the 2012 shared task on negation hosted by The First Joint Conference on Lexical and Computational Semantics (*SEM 2012) are comprised of the following negation annotated stories of Conan Doyle (CD): a training set of 3644 sentences drawn from *The Hound of the Baskervilles* (CDT), a development set of 787 sentences taken from *Wisteria Lodge* (CDD; we will refer to the combination of CDT and CDD as CDTD), and a held-out test set of 1089 sentences from *The Cardboard Box* and *The Red Circle* (CDE). Corpus statistics for CDT and CDD are reported in Table 4.4

	CDT	**CDD**
Sentences	3644	787
Sentences w/ negation	848	144
Cues	984	173
Scopes	887	168
Negated events	616	122
Tokens	65,450	13,566

Table 4.4: CD Corpus statistics for both the training and the development sets (Morante and Blanco 2012)

In these sets, the concept of negation scope extends on the one adopted in the BioScope corpus in several aspects: Negation cues are not part of the scope, morphological (affixal) cues are annotated and scopes can be discontinuous. Table 4.5 shows the most frequent cues in CDT and CDD.

Examples (4.6) and (4.7) below are examples of affixal negation and discontinuous scope, respectively.

(4.6) Since we have been so **un***fortunate* as to miss him [...]

(4.7) If he was in the hospital and yet **not** on the staff he could only have been a house-surgeon or a house-physician: little more than a senior student.

Moreover, the idea of scope itself is extendend with the concept of *negated events* (displayed in italicized characters in examples). These are the events and states within the negation scope that are semantically negated, generally closely located to their negating cue. By CD standards, "He does not eat meat." should be analyzed as "He does **not** *eat* meat.", where *eat* is the negated event. This annotation is only present in sentences that are factual, meaning that if the sentence

CDT		CDD	
not	359	not	42
no	229	no	35
un–	78	n't	20
n't	65	nothing	16
never	59	un–	16
nothing	55	never	11
without	24	without	7
–less	27	im–	6
in–	24	in–	5
im–	18	nor	4

Table 4.5: Frequency distribution for the ten most frequent negation cues in CDT and CDD.

were "He might **not** eat meat.", no part of the scope would be annotated, since the meat consumption is only potential. Example (4.7) has no negated events because the sentence expresses an hypothesis. Determiners, modifiers and auxiliaries are not marked as events; in multi-word phrases only the head is marked as an event.

Inspecting the dependency parser output for CDT and CDD reveals that, more often than not, there is a close, definite syntactic relation between cues and negated events. As we recall from Section 3.4, Stanford dependencies account for negation with a *neg* relation that links certain modifiers to the token they directly modify. Excluding those that are negated by affixes, infixes or suffixes (e.g. *interesting* in *uninteresting*) from CDT and CDD leaves us with 546 and 99 negated events respectively. In CDT, the number of events that the parser labels as heads in *neg*-labeled relations is 179; in CDD the count is 29. For CDTD, however, the set of modifiers that originate *neg* relations in the parsing model contains only the tokens *n't*, *not* and *never*, i.e. negation cues in the form of adverbial modifiers. If we take into account all the gold negation cues and recount the cue-event relations that are directly resolved by the parser via a single edge, the counts increase to 295 for CDT and 62 for CDD, i.e. 54% and 62% respectively.

Book/Chap.	Sentence #	Token #	Token	Lemma	PoS	Constituent	Negation		
baskervilles02	101	0	He	He	PRP	(S(NP*)	_	He	_
baskervilles02	101	1	never	never	RB	(ADVP*)	never	_	_
baskervilles02	101	2	returned	return	VBD	(VP*)	_	returned	returned
baskervilles02	101	3	. .	.	*)	_	_	_	

Figure 4.2: A sentence from CD, showing the tab-separated column format, the kind of information provided by each column and the annotations for cues, scope and negated events.

The Conan Doyle corpus is provided in the format depicted in Figure 4.2. In addition to information concerning negation cues, scopes and events, CD comes tokenized, lemmatized, PoS-tagged and parsed (Morante and Blanco 2012). Tokens are obtained with the tokenizer that is part of the LinGO English Resource Grammar,[3] lemmas and parts of speech with the Genia Tagger (Tsuruoka 2005) and parse trees with the Charniak and Johnson (2005) re-ranking parser.

Negation is represented in the three rightmost columns in Figure 4.2: cues are found in the first one, in-scope tokens in the second and negated events in the third. In case of multiple scopes, each additional cue originates its own triplet of columns, so that the length of each row depends on the number of scopes in the sentence. This annotation scheme allows for a more involved, multi-stratal representation of the relationship between cues, scopes and their overlaps.

4.4 Comparing corpora

Both PR and CD are valuable research tools in the context of the computational treatment of negation. From a language-domain perspective, they address the need for language resources annotated with negation information outside of the biomedical domain. Indeed, while not as marked, the difference in domain is true across these two corpora as well; PR consists of user-generated content from the web, while CD of fine *belle-epoque* prose. This difference can be observed in the frequency of cues like *n't* and *not*, the former being almost twice as frequent than the latter in PR, while it *not* occurs about 7 times as much as *n't* in CDTD. In addition, the dissimilarity in language register and style is quantifiable by looking at the difference in average sentence lengths across the two corpora, 8.07 tokens in PR and 17.83, more than double as much, for CDTD.

Most interestingly, the annotations themselves differ on a number of levels. Gold negation cues are not marked in PR; in CD cues are annotated, and they

[3] `http://moin.delph-in.net/`

can be made up of one or several words. Negation cues are annotated also on sub-word level; cues can be affixes, infixes and suffixes.

Furthermore, negation scopes are annotated following different paradigms. In PR, they are found almost solely to the right of a negation cue and they can be described as *minimal*, meaning that only the part of the sentence that is most prominently negated by a cue is annotated as its scope.

In CD, negation scopes are *maximal* and are comprised of full syntactic units. This means that scopes can occur on both sides of a negation cue and they can be discontinuous. In PR there is no instance of overlap between scopes; this is not the case in CD, where tokens can be assigned to the scope of more than one negation cue (see Figure 5.4 in Chapter 5 for an example).

The last, prominent difference between the two corpora is the annotation of negated events, an element that may be viewed as an even more minimal approach to negation scopes than the one adopted for the annotations in PR.

Chapter 5

System Description

Going from raw data to deep meaning analysis requires a number of steps. From an engineering point of view, writing a single piece of software that takes care of each of these steps is highly impractical for several reasons. First of all, the sheer amount of code that would have to be written if we were to develop each component from scratch would be gargantuan, and require the time of several master theses. Secondly, incorporating standard, state-of-the-art plugins that are publicly available[1] allows for easier result replication and contributes to foster a healthy, cooperative research community.

The external software packages and libraries that have been used to develop this system are the Natural Language Tool Kit (NLTK, Bird, Klein, and Loper 2009) for tokenizing, tagging and lemmatizing, Maltparser (Nivre et al. 2007) for dependency parsing and Wapiti (Lavergne, Cappé, and Yvon 2010) for sequence labeling. What we have built is an NLP pipeline that organizes the data flow, processing the data at each step so that it may serve as input for the next one. We have developed our own components for extracting negation-specific information, data formatting, post-processing and evaluation. The system is sewn together by parametrizable scripts that allow for comfortable experimentation across different system configurations.

Figure 5.1 is a diagram of this pipeline. First, the raw corpus is processed and enriched with lexical and syntactic information; then, the data is converted to a format that fits the specifications of the sequence-labeler and additional contextual features are extracted. The data thus structured is fed to the learning algorithm, that returns a model we can use to label unseen data. Finally, the output of the labeler is post-processed to combine the labels into full scopes before being evaluated.

[1] And often, thankfully, open-source.

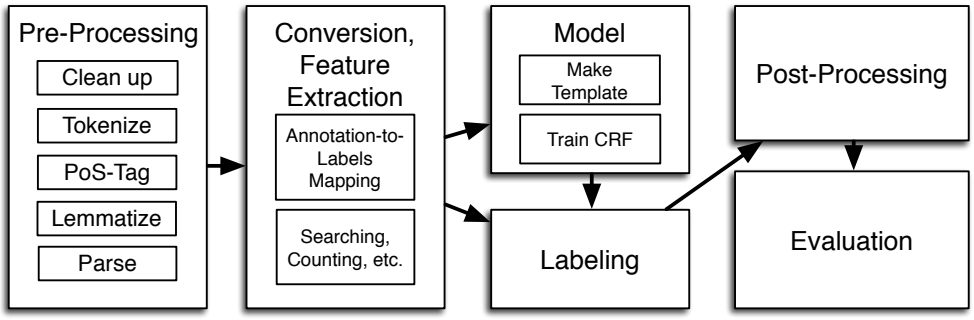

Figure 5.1: Pipeline diagram that shows the different processing steps in our system.

The rest of this chapter will first detail the pre-processing step in Section 5.1, with special attention to the work done with the 'rawest' of our two corpora, the Product Review Corpus. Section 5.2 presents the work done in terms of feature encoding and extraction, while Section 5.3 details the model internal representation. Lastly, we explain post-processing in Section 5.4, leaving evaluation methods and description of the experimental setup for the next chapter.

5.1 Pre-processing

The first box in Figure 5.1 depicts the pipeline-within-the-pipeline that is pre-processing. After "cleaning" the text from non-linguistic information,[2] the first important NLP process applied to the raw text is *tokenization*, or the breaking up of text "[...] into distinct, meaningful units" (Kaplan 2005). While the aim of tokenization is fairly easy to understand, the process itself comes with some challenges. Punctuation is the first that comes to mind, with the ambiguity of periods being used both for ending sentences and create abbreviations: for instance, the abbreviation 'chap.' for *chapter* looks just the same as the word *chap* located at the end of a sentence. Ideally, we would retrieve the former as one token, `["chap."]`, and the latter as `["chap", "."]`.

Contracted forms in English are also problematic. Table 5.1 shows 5 possible tokenization approaches for *can't*, where it is either kept as it is or split.

[2]For instance html tags that we know we don't want to analyze, or handling the conversion of code like '`å`' into '`å`'.

Tokenization variants	
can't	`["can't"]`
	`["ca", "n't"]`
	`["can", "n't"]`
	`["can", "not"]`

Table 5.1: Different tokenization strategies for *can't*.

The three split versions include one that simply separates `"ca"` and `"n't"`, one where only `"can"` is expanded and one with the full version of both `"can"` and `"not"`; one might argue for the validity of each of these tokenization strategies. The tokenizer used for this work complies with the standards of the Penn Treebank (PTB, Marcus, Marcinkiewicz, and Santorini 1993), a corpus so widely used in NLP that it has become the *de-facto* standard for tokenization and part-of-speech nomenclature.[3]

The reason behind this choice is simple: both the PoS-tagger and the parser used after the tokenizer in the pipeline are trained on the PTB, so in order to obtain usable tag sequences and graphs, the data has to be as compatible as possible with the next component.

The next pre-processing step in the pipeline is to tag the tokens with parts of speech — lexical categories that group words according to certain properties. Nouns, adjectives, adverbs, determiners, pronouns and verbs with their sub-categories and inflections are both used on their own to design features and required for syntactic parsing. Again, in order to acquire healthy additional information, it is vital that these annotations are compatible across components. We utilize the default n-gram tagger in NLTK, also trained on the PTB, which uses the same HMM tagging technique described in Chapter 2.

With help from the PoS tags we obtain the base form of tokens, or *lemma*, via NLTK's Wordnet lemmatizer. The level of generalization provided by lemma of a word is somewhere between a token and its PoS tag; inflected forms of verbs like *is* and *are* are reduced to their infinitive form (in this case, *be*) and plural nouns like *skies* to their singular form (e.g. *sky*).

Finally, we parse the data using Maltparser, the robust, open-source dependency parser that was presented in Section 3.3. We used the available model for English, which is pre-trained on a conversion of sections 2-21 of the Wall Street

[3] We note that the PTB approach to tokenization is still debated and challenged, most recently in Dridan and Oepen (2012).

Journal section of the Penn Treebank to Stanford dependencies augmented with 4000 questions from the QuestionBank, another syntactically annotated corpus.

5.1.1 Reformatting and enriching the Product Review corpus

The tokenization, PoS-tagging and lemmatization that were just described apply only to the Product Review corpus; as we have seen in Chapter 4, the Conan Doyle corpus comes tokenized, tagged and lemmatized. the format of PR is basically raw text with xml-like tags that bound sentences and scopes; to enable external, standard evaluation via the *SEM shared task evaluation script (discussed in Chapter 6), we ported the corpus to the same format of CD.

As we recall, one major difference between the two corpora is the fact that PR is not annotated with gold negation cues. To amend this shortcoming, we inspect and semi-automatically annotate the corpus ourselves with the following strategy. With a python script, we inspect the first out-of-scope token to the left of each scope, matching it to the same lexicon in Table 4.1 used by Councill, McDonald, and Velikovich (2010); this search does not yield a match in only 5 out of 730 scopes. In one of these 5 sentences, the cue is to the right of the scope:

(5.1) [...] but subtle it ai**n't**.

In two sentences, the cue is simply not immediately preceding the scope:

(5.2) [...] heart-of-gold Alistare is **not**, a ghost to give up so easily.

(5.3) [...] even if they do**n't** (or couldn't) happen to us.

We consider the remaining two sentences to be annotation errors, where the human annotators confused negation with speculation:

(5.4) [...] who could 've lived [...]

(5.5) [...] who could have been very good [...]

We then proceeded to hand-annotate the three long-distance negation cues in Examples 5.1, 5.2 and 5.3, and automatically annotate the rest of the corpus, while discarding the two sentences annotated with speculation.

The first block in Figure 5.2 shows a native PR sentence, the second one is the CD-like version with gold cues. The third block is the representation that is presented to the labeler, which contains the additional features presented in the next section.

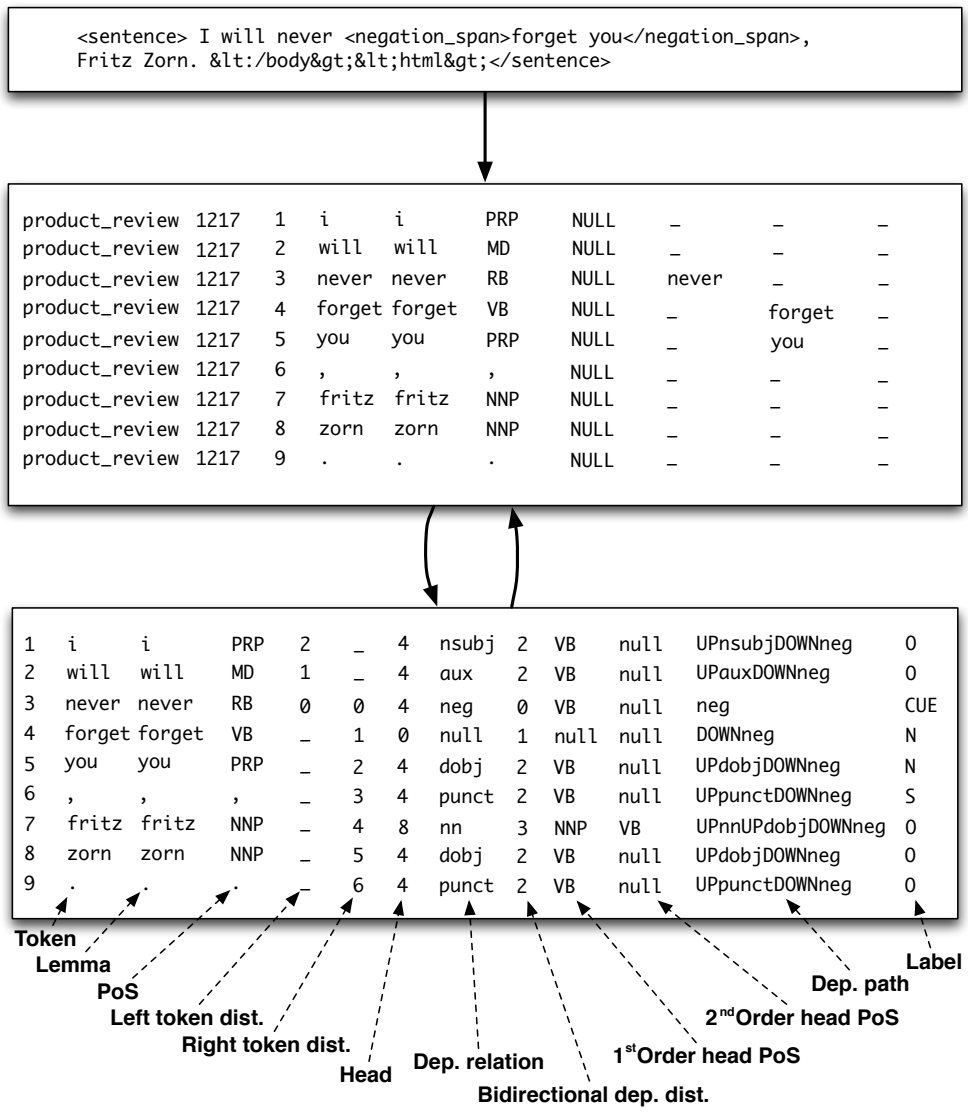

Figure 5.2: This figure shows three different versions of a sentence in PR. The first block is the sentence as it is in the raw corpus, while the second shows how the sentence is formatted in the same style as CD. The third block is the format used in the model, containing the extracted features and the sequence of labels. After labeling, the sentence is converted from the third format to the second, pairing scopes (and events, in CD) to their cues and enabling evaluation via *SEM shared task evaluation script.

5.2 Feature engineering

In order to create a model that describes negation based on the training data, we instruct our training algorithm to record certain patterns at each state of each sequence of tokens. Rather than manually encoding specific feature patterns, say the token *up* with its PoS *RP* preceded by a *VBZ*-tagged token, we define feature *macros*, generalized descriptions of explicit items in the data set and their combinations. A feature macro for the instance we just described simply asks the CRF learner to retrieve, for each state in the sequence, the current token with the current and the preceding part of speech. During training, the learning algorithm records each instance matching this generalized description, and assigns it a weight according to how well it describes the current label.

Features	
Lexical	Token Lemma PoS unigram Forward token bigram and trigram Backward token bigram and trigram Forward PoS trigram Backward PoS trigram Lexicalized PoS Forward Lexicalized PoS bigram Backward Lexicalized PoS bigram
Syntactic	Constituent (In CD) Dependency relation First order head PoS Second order head PoS Lexicalized dependency relation PoS-disambiguated dependency relation
Cue-dependent	Left Token distance Right Token distance Directed dependency distance Bidirectional dependency distance Dependency path Lexicalized dependency path

Table 5.2: List of features used to train the CRF models.

With the information available from the gold annotations, the preprocessing and the explicit encoding of additional contextual and syntactic information, we record a number of said features macros, listed in Table 5.2, and collect them in the *template* files that are interpreted by Wapiti, the CRF package.

The features presented in Table 5.2 are subdivided in *Lexical*, *syntactic* and *cue-dependent*; the latter are contextual to negation, which means they rely on the presence of an explicit negation cue. The sequence labeler operates via the feature template on the version of the dataset shown in the third block of Figure 5.2. Some of the features are extracted and explicitly encoded in each column in the dataset, while others are *expanded* in the feature template.

Our features are developed in a fairly austere fashion, starting from the feature-set from Councill, McDonald, and Velikovich (2010) and adding more specialized ones as we empirically assess that they were beneficial to performance. Our approach is more aggressive than Councill, McDonald, and Velikovich (2010), and extends their feature-set in several ways; for example rather than recording surface bigrams and trigrams in one direction only, we look in both directions, doing the same for PoS tags. In addition, parts of speech and dependency relations are both combined and lexicalized. All our models are produced using the default settings in Wapiti.

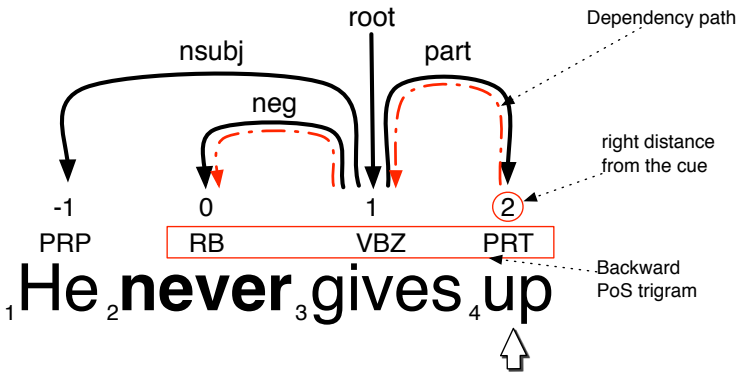

Figure 5.3: In this example sentence showing the active *right token distance* and *backward token trigram* features when the currently processed token is *up*. Looking at the accompanying dependency graph, we see that the active *dependency path* feature is $\uparrow\ part \downarrow\ neg$.

Figure 5.3 will help us make a simplified example of how certain features can guide the classifier to make a label assignment, which aims at clarifying why modeling the language of the corpus in detail is a defensible idea. The delexicalized trigram RB / VBZ / PRT is a pattern that matches contexts like

always sleeps in, *sometimes slows down* and the one shown in the example, *never gives up*. While most RB — in English, adverbs — observed in the corpus are not negation cues, the converse is also true, i.e. most negation cues are adverbs. This means that the RB / VBZ collocation has a stronger association with negation than, say, PRP[4] / VBZ.

Imagine that our labeler has to assign label probabilities to *gives* in Figure 5.3; it is reasonable to expect a very high probability for an in-scope label, since in the vast majority of examples in our corpus, the word immediately following a negation cue is observed to be in-scope. There is however even more evidence that *gives* should be in scope, since in *all* the examples in the corpus where RB is a negation cue and is followed by VBZ, the latter is in-scope.

Now, imagine that *up* is the currently processed token by the labeler. For the sake of the example, assume that tokens with right token distance from a cue equal to 2 are observed to have an equal chance of being in or out of scope. While the backward trigram RB / VBZ / PRT (remember, the current token is *up* and we are looking 2 steps back) is more strongly associated with negation than PRP / VBZ / PRT, on its own it still yields a higher probability for an out-of-scope label, because adverbs are for the most part not negation cues. The backward bigram VBZ / PRT is also not helpful in guiding the classifier to make the right decision. Assuming that the lexical trigram *never gives up* has not been observed, the model needs more information to assign higher probability to the in-scope label, and the next logical place to look for more information is syntax.

5.2.1 Dependency features

The features extracted via the dependency graphs aim at modeling the syntactic relationship between each token and the closest negation cue. Parsing each sentence with Maltparser returns dependency graphs, non-linear representations that can be explored to find more general traits that characterize in-scope tokens.

With the token indices as unambiguous identifiers and the head relations being incoming edges, we represent the dependency graph as a set V of vertices and two different sets of edges, E and E′ — the former containing only the directed edges and the latter containing also the reversed.

For Figure 5.3, we have:

- $V = \{1, 2, 3, 4\}$
- $E = \{\langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 3, 4 \rangle\}$
- $E' = \{\langle 3, 1 \rangle, \langle 3, 2 \rangle, \langle 3, 4 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle, \langle 4, 3 \rangle\}$

[4]Personal Pronoun, for example *He*.

From the graph G = {V, E} we extract the shortest path from the *head* of the negation cue to all other vertices in the graph; we start from the head of a cue because, being directed acyclic graphs essentially trees, negation cues are very often found in the leaves. This would yield no path from the cue to anywhere in the tree.

From G′ = {V, E′} we extract the shortest path from the negation to all other vertices. If there is more than one negation cue in the sentence, the relevant cue is the lexically closest one to a token with no intermediate punctuation. Shortest paths are computed using an unweighted implementation of the Dijkstra algorithm (Dijkstra 1959).

The number of nodes in each of these two different shortest paths is encoded as a dependency distance feature; in the case of the directed representation, we record -1 in case no path is found. While both distance measures on their own contributed positively to the model during development, the dependency distance calculated on the bidirectional representation proved to be more effective than the other when not used together; using them in conjunction seemed to confuse the model, thus the final model utilizes only the bidirectional distance.

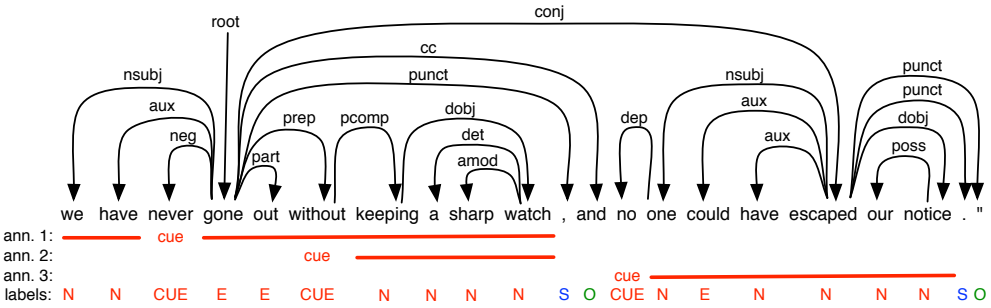

Figure 5.4: A sentence from the CD corpus showing a dependency graph and the annotation-to-label conversion.

We furthermore use these shortest paths to record the *Dependency Graph Path* as a feature. This feature was inspired by the Parse Tree Path feature presented in Gildea and Jurafsky (2002) in the context of Semantic Role Labeling. It represents the path traversed from each token to the cue, encoding both the dependency relations and the direction of the arc that is traversed: for instance, the syntactic relation between *our* and *no* in Figure 5.4 is described as
$\uparrow\ poss \uparrow\ dobj \downarrow\ nsubj \downarrow\ det$.

Recall our thought experiment in the previous section, which, without syntactic information, ended with an incorrect label assignment for the token *up*

in Figure 5.3; if the active feature $\uparrow \ part \downarrow \ neg$ has been observed only in conjunction with an in-scope label, the model would be likely to assign a high probability for *up* to be in-scope, eventually leading to the correct label assignment.

Finally, like Councill, McDonald, and Velikovich (2010), we also encode the PoS of the first and second order syntactic head of each token. For the token *no* in Figure 5.4, for instance, we record the PoS of *one* and *escaped*, respectively. Unless explicitly stated, in all CD configurations the available constituent information from the parse tree is added to the model.

5.3 Model-internal representation and labeling

Councill, McDonald, and Velikovich (2010) use two labels in their classifier. Since different kinds of label transitions do affect the decision of the sequence labeler, we attempt to capture the behavior of specific tokens within the mechanics of negation not only with feature modeling, but also with a finer-grained label set. Because of the difference in information available in the two corpora, we subdivide this section in two parts, starting with CD.

5.3.1 Conan Doyle

The token-wise annotations in the CD corpus contain multiple layers of information. Tokens may or may not be negation cues and they can be either in or out of scope; in-scope tokens may or may not be negated events, and are associated with each of the cues they are negated by. Moreover, scopes may be (partially) overlapping, as in Figure 5.4, where the scope of *without* is contained within the scope of *never*. We convert this representation internally by assigning one of six labels to each token: O, CUE, MCUE, N, E and S, for out-of-scope, cue, morphological (affixal) cue, in-scope, event and negation stop respectively.

The CUE, O, N and E labels parallel the IOB chunking paradigm and are eventually translated in the final annotations by our post-processing component. MCUE and S extend the label set to account for the specific behavior of the tokens they are associated with. The rationale behind the separation of cues in two classes is that, since cues across different grammatical classes generate different kinds of scopes, the labeler could benefit from knowing the difference.[5]

Table 5.3 presents the frequency distribution of PoS-tags over the different cue types in CDTD and shows that, unsurprisingly, the majority class for morphological cues is adjectives, which typically generate different scope patterns

[5]This grouping could also have been done by encoding extra features.

PoS	# S	PoS	# MCUE	PoS	# CUE
punctuation	1492	JJ	268	RB	1026
CC	52	RB	28	DT	296
IN + TO	46	NN	16	NN	146
RB	38	NN	4	UH	118
PRP	32	IN	2	IN	64
rest	118	rest	~	rest	38

Table 5.3: Frequency distribution of parts of speech over the S, MCUE and CUE labels in CDTD.

compared to the majority class for standard cues. The S label is applied to special instances of out-of-scope tokens. An S-labeled token is defined as the first non-cue, out-of-scope token to the right of one labeled with N. As Table 5.3 shows, the S label targets mostly punctuation, which is the case in Figure 5.4.

After some experimentation with joint labeling of scopes and events, we opted for separation of the two models, and hence train separate models for the two tasks of scope resolution and event detection. In the model for scopes, all event labels are switched to in-scope; conversely, in-scope tokens become out-of-scope tokens in the event model. As described in section 4.3, only scopes in factual context contain an event. Hence, the predictions provided by the model for events serve a double purpose: finding the negated token in a sentence and deciding whether a sentence is factual or not. The outputs of the two classifiers are merged during post-processing.

5.3.2 Product Review

The conversion into label sequences is more straightforward for PR, which does not contain negated events or overlapping scopes. The representation is essentially the same as for CD, with the difference that there is no MCUE label, since there are no morphological cues in the corpus. In order to have label sets of equal granularity across corpora, we group all cues tagged as adjectives or determiners into a CUEAD label.

5.4 Post-processing

A simple, heuristics-based algorithm was applied to the output of the labelers in order to pair each in-scope token to its negation cue(s) and determine overlaps. Our algorithm works by first determining the overlaps among negation cues. Cue A negates cue B if the following conditions are met:

- B is to the right of A.
- There are no tokens labeled with S between A and B.
- Token distance between A and B does not exceed 10.

In the example in Figure 5.4, the overlapping condition holds for *never* and *without* but not for *without* and *no*, because of the punctuation between them. The token distance threshold of 10 was determined empirically on CDT. In order to assign in-scope tokens to their respective cue, tokens labeled with N are treated as follows:

- Assign each token T to the closest negation cue A with no S-labeled tokens or punctuation separating it from T.
- If A was found to be negated by cue B, assign T to B as well.
- If T is labeled with E by the event classifier, mark it as an event.

This algorithm yields the correct annotations for the example in Figure 5.4; when applied to label sequences originating from the gold scopes in CDD, the reported F_1 is 95%. We note that this loss of information could have been avoided by presenting the CRF with a version of a sentence for each negation cue. Then, when labeling new sentences, the model could be applied repeatedly (based on the number of cues provided by the cue detection system). However, training with multiple instances of the same sentence could result in a dilution of the evidence needed for scope labeling. Preliminary labeling results using this approach show slight improvements, but due to time constraints this remains to be investigated properly in future work.

5.5 In short

Our system is a complex NLP pipeline that starts with tokenization of raw text annotated with the scope of negation and ends with *SEM 2012-styled, automatically annotated negation scopes. We enrich the annotation of the Product

Review Corpus with gold cues, which are annotated in a semi-automatic fashion. The multi-stratal *SEM annotations are converted into simple sequences of labels from a fine-grained set, and we extract detailed lexical, syntactic and contextual features to create CRF models for negation. At the end of the pipeline, a simple algorithm restores the flat sequence labels into full scopes.

Chapter 6

Evaluation & Experimental Setup

An intuitive, trivial way of evaluating our outputs is to calculate the percentage of correct labels. Taking into account the very lopsided label ratio, however, suggests that this might not be such a good idea. Labeling all PR tokens as out of scope would yield 95% correct labels; this is both an impractical baseline, where minuscule improvements can actually conceal meaningful differences, and a possibly misleading result.

This chapter introduces the more informative scores used in this work and other important, methodological aspects of our experimental setup.

6.1 Precision, Recall and F_1 score

Traditional evaluation of NLP systems is most often done in terms of Precision (P), Recall (R) and F_1 Score, evaluation measures that were initially introduced to evaluate Information Retrieval (IR) systems (Manning and Schütze 1999). An IR task usually has some target information (named entities such as *Google*, say) that has to be retrieved. With the universe divided into target and non-target items, an IR system has 2 different ways to be wrong and another two to be right; the former are known as false negatives and positives (*fn* and *fp*), while the latter are true negatives and positives (*tn* and *tp*). The Venn diagram in Figure 6.1 shows the relation between the whole data-set, the the gold instances (our target items, the elements the system is looking for) and the ones retrieved by the system. The space labeled with *tn* represents irrelevant items that are not retrieved by the system, while the one labeled with *tp* corresponds to the sought-after items that were retrieved. The *fp*-space represents retrieved instances of items that are actually irrelevant, while the *fn*-space covers relevant items that the system failed to retrieve.

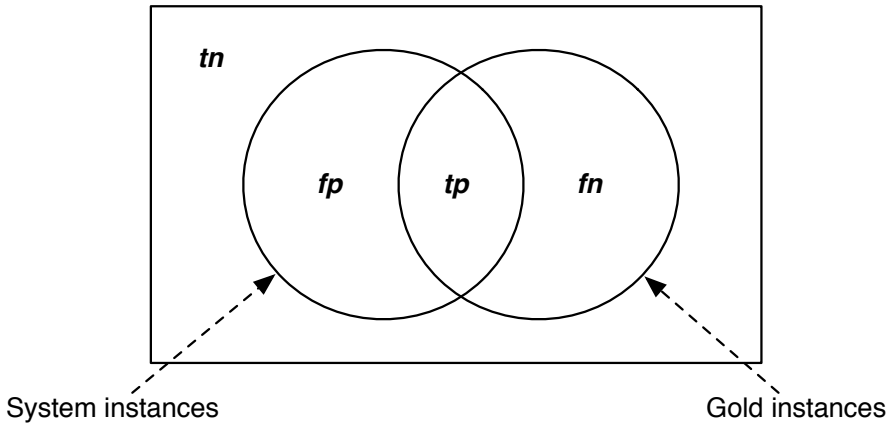

Figure 6.1: A Venn diagram with system outputs and gold instances dividing the space in true positive (tp), true negative (tn), false positive (fp) and false negative (fn) predictions.

Because they do not take into account true negatives, Precision and Recall measure the performance of the system in terms of the target class only. Precision, which rewards the performance of the system in terms of how good it is in not generating false positives, is computed as follows:

$$\mathrm{P} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}} \tag{6.1}$$

Since the result is a real number between 0 and 1, Precision can be viewed as the probability that picking an item at random among the set retrieved by the system will yield a relevant item, i.e. one that is in the set of gold instances.

Recall on the other hand, rewards the performance of the system according to how many of the gold instances are retrieved. It is calculated like this:

$$\mathrm{R} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}} \tag{6.2}$$

Like Precision, Recall can be viewed as a probability — in this case, the probability that a randomly selected element in the gold instances is found by the system.

On its own, Precision measures the reliability of a system's predictions, while Recall measures the robustness of the system. Looking at each of these scores

in isolation can be potentially misleading; perfect Precision can, in theory, be obtained by instructing a system not to retrieve anything at all, while in a conceptually simple IR task (e.g. retrieving all non-English words in a text), perfect Recall can be obtained by retrieving everything. For these reasons, the harmonic mean of P and R, called F_1 score, is generally used to measure the overall performance of a system. The equation is the following:

$$F_1 = \frac{2PR}{P + R} \tag{6.3}$$

As discussed in the introduction, calculating accuracy over all labels is probably not a good idea for our scope-resolution task; computing the F_1 score for the target label only is a better approach. In this case, a *tp* is a gold in-scope token labeled as in-scope by the system, an *fp* is an out-of-scope token labeled as in-scope and an *fn* one that is incorrectly labeled to be out of scope. This approach, however, is still not one that takes into account *consecutive* chunks of tokens nor their (often overlapping) relation to cues.

Applying IR evaluation metrics to scope resolution is a choice that needs to be justified: in the next two sections we will introduce PCS and the evaluation measures produced by the 2012 *SEM shared task evaluation script that are relevant for this work.

6.2 The PCS score

Morante and Daelemans (2009) introduce the *Percentage of Correct Scopes* (PCS) measures as a stricter way to evaluate scope resolution systems than F_1 score over tokens. With #PS being the number of correct scopes produced by the system and #S the number of gold scopes, PCS can be expressed with:

$$PCS = \frac{\#PS}{\#S} \tag{6.4}$$

In the diagram in Figure 6.1, #S is the gold instances-set (i.e. *tp* + *fn*), while #PS represents true positives. Since false positives are not taken into account, PCS is a recall measure. In Morante and Daelemans (2009), a correct scope takes into account the full annotation of of a bioscope-scope, i.e. "A scope is correct if all the tokens in the sentence have been assigned the correct scope class for a specific negation signal" (Morante and Daelemans 2009). Since there are no gold cue annotations in the Product Review Corpus, Councill, McDonald, and Velikovich (2010) relax this definition to some extent, calculating PCS as the "[...] number of correct spans divided by the number of true spans". Figure 6.2 shows an example of the latter, cue-less interpretation, with the top instance

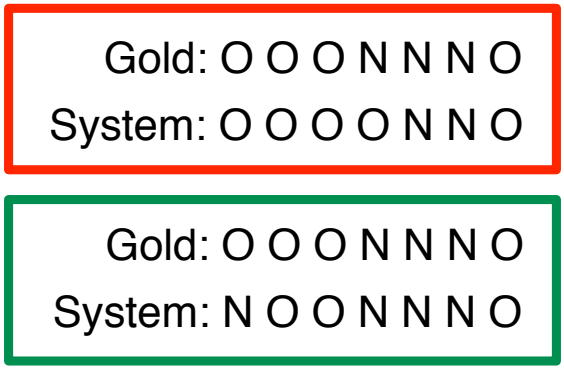

Figure 6.2: A practical example of how incorrect (top) and correct (bottom) systems outputs are counted to compute the PCS score. The top instance is incorrect because the sequence of tokens representing the scope is in the gold data is not a perfect match in the system output; the bottom instance is considered correct because the whole scope is retrieved by the system, although one out-of-scope token has been mislabeled.

being counted as incorrect and the bottom one as correct. When comparing our results directly to Councill, McDonald, and Velikovich (2010), we will adopt this PCS interpretation.

6.3 The *SEM shared task evaluation script

The results from the next chapter are predominantly obtained using the evaluation script from the 2012 *SEM shared task, which covers not only scopes but a number of aspects of negation (Morante and Blanco 2012). Most of the scores produced by the script follow the F_1 measure paradigm.

A true positive scope is a one that is identical in the system output and in the gold corpus, while a false positive is one that originates from a false positive negation cue. A false negative scope is one that contains either false positive or false negative scope tokens. P, R and F_1 are computed as illustrated in Section 6.1.

For negated events, the *tp* / *fp* / *fn* set is computed with the standard, IR-like strategy explained in Section 6.1; a false positive is a retrieved event that is not annotated as such in gold, and a false negative a gold event that has not been

retrieved.

The score used to produce the final ranking of the shared task is %CNS (Percentage of Correct Negation Sentences), the strictest evaluation measure reported by the script. A correct negation sentence is one that is retrieved with all of scopes, cues and events being identical to their hand-annotated counterparts; %CNS is computed dividing the total number of these instances with the total number of sentences containing at least one scope, making it a recall measure.

6.4 Experimental setup and settings

Before diving into the experiments, some details regarding the actual setup are in order.

Significance testing

Statistical significance for differences in performance on equal data sets is assessed using a two-tailed sign test applied to output pairs of scope or event predictions. This is a standard non-parametric test for paired samples (Velldal, Øvrelid, Read, and Oepen 2012), which compares the differences in the predictions of two systems, the null hypothesis being that the differences between the two outputs are due to chance. We will assume a threshold, or alpha level, of 0.05; a test returning a p value above this threshold confirms the null hypothesis.

Training and testing

In our experiments, the training and testing regimen is slightly different across the two corpora. In order to maximize the size of the Product Review corpus, all experiments in Chapter 7 that involve this corpus are cross validation experiments.

This technique is often used in similar supervised classification tasks, and it consists in splitting a corpus in n non-overlapping folds so that a model can be trained on $n - 1$ folds and tested on the remaining one, repeating the experiment n times exchanging the test fold with one from the training corpus each time. A visual example with $n = 3$ is pictured in Figure 6.3.

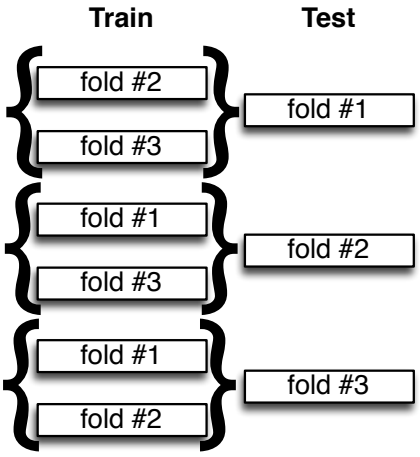

Figure 6.3: n-fold cross validation example with $n = 3$.

The Product Review experiments are performed with ten folds, and the reported results are the averages of the results for each fold. For instance, with R being the reported recall for a Product Review experiment and r_i the recall score for the ith fold, we have:

$$R = \frac{1}{10} \sum_{i=1}^{10} r_i \tag{6.5}$$

Experiments on the Conan Doyle corpus abide by the standards of the 2012 *SEM shared task; unless explicitly stated, we train using the designated training corpus and test on the development corpus. While CD comes with a held-out set, focusing on CDD allows us to perform an error analysis without incurring in methodological issues.

Settings

As already stated in the previous chapter, dependency parses are obtained using the default configuration of Maltparser, which utilizes an implementation of the parsing algorithm described in 3.3, and the linear SVM version of the pre-trained Model for English.[1] Wapiti, the sequence labeler, is always run with its default

[1]Both the parser and the model are available from `http://www.maltparser.org`.

settings.[2] Experimentation with different parser and labeler configurations and algorithms has not been undertaken because of the time constraints.

[2]Wapiti can be obtained from `http://wapiti.limsi.fr`.

Chapter 7

Experiments

With our system and evaluation metrics explained, we present a series of experiments performed with the available corpora. In order to be perfectly clear as to what the coming results are supposed to communicate to the reader, we formulate three main questions that our experiments are designed to answer.

(7.1) Are features extracted via dependency graphs helpful?

(7.2) Does our fine-grained set of labels affect the performance of the classifier?

(7.3) Are there any noteworthy differences in performance across the two different domains and annotation guidelines?

This chapter is structured as follows: after introducing baselines in Section 7.1, we perform internal evaluation of our system in Sections 7.2 and 7.3 and take a close look at some errors in 7.4. Lastly, we compare our efforts to other work in Section 7.5.

7.1 Baseline systems

The first order of business is to establish a *baseline*, a minimum starting point which we can use for comparisons. A way to accomplish this is to the devise the simplest possible algorithm to solve a certain task, apply it to the data and collect the results; our approach is to consider all tokens in a sentence to be in-scope whenever we find a negation cue, using punctuation as a delimiter. With the differences between our two sets of data in mind,[1] we consider both sides

[1] As already established in Chapter 4: scopes in CD can be discontinuous and can propagate to both sides of a negation cue, whereas PR allows only continuous scopes that are on the right of negation cues, save for one instance.

of the negation cue for CD while for PR we start from a cue and stop at the first punctuation mark to its right; Examples 7.4 and 7.5 show PR and CD style annotations for the same sentence.

(7.4) America is **not** the world.

(7.5) America is **not** the world.

The results for scope resolution using this simple method are reported in Table 7.1 under *Cue To delimiters*. As a form of internal baseline, we present the results of systems that do not take advantage of dependency-based features and use the minimal set of labels, namely N, O and CUE, paralleling the standard IOB setup.

Dataset	Method	Prec	Rec	F_1
PR	Cue to delimiters	22.10	99.76	36.18
	{CUE, N, O}, no dep	100.0	65.55	79.19
CDD	Cue to delimiters	68.50	86.65	76.51
	{CUE, N, O}, no dep	100.0	63.69	77.82

Table 7.1: Gold-cues scope-resolution baselines for both corpora compared with a simple configuration of our system with 3 labels and no dependency features.

Table 7.1 provides two very important insights. For PR, the simplest configuration of our machine for scope resolution outperforms the simplest conceivable algorithm by a rather large margin. For CD, the *cue-to-delimiters* heuristics yields a very strong baseline which our system barely surpasses.[2] The reason for this very marked baseline difference across corpora is fairly intuitive, and lies in the different nature of the annotations — minimal scope for PR and maximal for CD.[3] The baseline classifier fails to correctly reproduce the scope of *not* in Example 7.6 from PR; re-annotating the same sentence following the CD standards as in Example 7.7, the CD baseline heuristics correctly resolve the scope.

(7.6) I just do**n't** care about August Wilson's writing.

(7.7) I just do**n't** care about August Wilson's writing.

[2]Note, however, that the difference is statistically significant.

[3]See Chapter 4 for a detailed description of the corpora.

7.2 Product Review

The experiments conducted using PR are ten-fold cross validation experiments. In order to make the most out of the small corpus, we split it in ten non-overlapping folds and run ten experiments, each using a different fold for testing and the remaining 9 for training.

Table 7.2 shows the averages from the Precision, Recall and F_1 for perfect scopes reported by the *SEM2012 evaluation script for each test-fold. While negation cue classification is an essential part of resolving negation, it is also a task with its own challenges and techniques. In order to isolate the performance of our scope-resolving system, we run this set of experiments using the gold cues extracted as described in Section 5.1.1.

	Configuration	**Prec**	**Rec**	**F_1**
(1)	{CUE, N, O}, no dep	100.0	65.55	79.19
(2)	{CUE, N, O}, dep	100.0	66.65	79.82
(3)	{CUE, N, O, CUEAD, S}, dep	100.0	67.19	**80.22**

Table 7.2: Scope-resolution experiments with gold cues for PR. (1) is the basic configuration of our system with 3 labels and no dependencies, (2) adds dependency features to the model and (3) takes advantage of both dependency features and 2 additional labels.

To answer Question 7.1[4] we compare the 79.19 F_1 score from our internal baseline model in Row (1), Table 7.2, that is no dependency and minimal {N, O, CUE} label-set, with the system configuration using dependency features and the same label-set, F_1 score 79.82 reported in Row (2). The small, statistically insignificant at α = 0.05, improvement margin of 0.77 percentage points observed comparing F_1 scores might not justify the time needed to parse the PR corpus and extract the additional features. We attribute this to two main factors. First, as already discussed and exemplified in Chapter 4, user generated content is hard to parse, which means that the graphs we are extracting features on might not be as reliable as we expect them to be. Secondly, the nature of the annotation in PR, with minimal, continuous scopes, might make deeper syntactic analysis superfluous.

Suspecting that the n-gram features and PoS do most of the job when it comes to modeling the relationship between cues and their scopes in PR we try to isolate their effects in another experiment. We train three models: (A) using only

[4] Are features extracted via dependency graphs helpful?

unigram tokens and token distance from the cue, (B) where we take advantage of PoS and n-grams combinations and (C) that uses lexical and dependency features exclusively. The results in Table 7.3 seem to confirm our hypothesis. We see that (B) and (C) yield nearly identical results, with F_1 scores of 79.19 and 79.25 respectively; performance drops to 61.65 ablating both dependency features and PoS/n-gram collocations, while their combined effects yield the already mentioned 80.22 F_1 score.

	Configuration	**Prec**	**Rec**	**F_1**
(A)	tokens & distance	100.0	44.91	61.65
(B)	(A), PoS & collocations	100.0	65.55	79.19
(C)	(A) & dependency features	100.0	65.64	79.25

Table 7.3: Feature ablation experiments for PR. (A) uses only token unigrams and left/right token distance from a negation cue, (B) adds the token and PoS patterns and (C) adds the features derived from the dependencies.

To answer question 7.2, on the utility of our fine-grained set of labels, we return to Table 7.2, comparing the 80.22 F_1 score in Row (3), our best performing classifier, to (2). While we observe a small, 0.4 percentage points improvement, it is not statistically significant, so it seems likely that the fine-grained set of labels does not improve performance on PR.

7.3 Conan Doyle

Configuration	**Prec**	**Rec**	**F_1**
(1) {CUE, N, O}, no dep	100.0	63.69	77.82
(2) {CUE, N, O}, dep	100.0	64.88	78.70
(3) {CUE, N, O, MCUE, S}, dep	100.0	67.26	**80.43**

Table 7.4: Scope resolution experiments for CDD with gold cues. (1) is the basic configuration of our system with 3 labels and no dependencies, (2) adds dependency features to the model and (3) takes advantage of both dependency features and 2 additional labels.

Following the standard established by the *SEM shared task, we train our system on CDT and test on CDD rather than performing 10-fold cross validation

on CDTD. [5] For the same reasons as for PR, we take advantage of gold-cues to concentrate on our scope-resolution component. The results from Table 7.4 show that the performance of the model trained on the bigger label-set is more marked compared to its PR counterpart, with a significant improvement measuring almost 2 whole percentage points between (2) and (3). While the impact of dependency features on scopes alone appears to be marginal, with less than a 1 percentage point improvement comparing (1) and (2), we report results that validate our graph-traversing efforts in Table 7.6 and discuss them in the next section. In contrast to the parallel PR experiment, however, all improvements are found to be statistically significant.

We repeat the same feature ablation experiment we did for PR on CD, in order to assess the degree of overlap between the PoS / lexical patterns and dependency features. The results shown in Table 7.5 are comparable to those in Section 7.2, though the difference between (B) and (C) is more definite.

	Configuration	**Prec**	**Rec**	**F_1**
(A)	tokens & distance	95.16	35.12	61.65
(B)	(A), PoS & collocations	100.0	64.29	78.26
(C)	(A) & dependency features	100.0	66.07	79.57

Table 7.5: Feature ablation experiments for CDD. (A) uses only token unigrams and left/right token distance from a negation cue, (B) adds the token and PoS patterns and (C) adds the features derived from the dependencies.

7.3.1 Event finding

The annotation of negated events and states within negation scopes in the CD corpus is minimal in nature; while CD negation scopes have been found to align to a large degree with entire syntactic units (Read, Velldal, Øvrelid, and Oepen 2012), the link between cues and negated events is more specific, often very close to the head-dependent relation between the two; as already noted in Section 4.3, most negated events in CDTD (not counting those negated by a morpheme within the same token) are in fact directly linked to the cue that negates them by a dependency arc.

As already mentioned in Chapter 4, negated events are minimal and do not include determiners, modifiers or auxiliaries. This results in cue and events being relatively often closer, if not consecutive, in terms of dependency distance than

[5] Results on CDE are provided in the comparison with related work in Section 7.5.2

they are lexically; in CDTD, 89% of negated events are at most two dependencies away from each other. This can be seen in the following examples from the annotation guidelines (Morante, Schrauwen and Daelemans 2011):

(7.8) There is **no** sick *man*.

(7.9) [...] we *can* get **nothing** out of him.

The results of resolving negated events in CDD benefit from the larger label-set and dependency features to a larger degree than they do for scopes; Table 7.6 shows an 8.27 percentage point improvement taking advantage of the dependency features and an additional 2.17 points improvement when adding the extra labels. Both improvements are deemed statistically significant by the sign test.

Configuration	Prec	Rec	F_1	
(1)	{CUE, E, O}, no dep	100.0	62.61	70.24
(2)	{CUE, E, O}, dep	81.55	75.68	78.51
(3)	{CUE, E, O, MCUE, S}, dep	83.65	77.68	**80.55**

Table 7.6: Negated event resolution experiments for CDD with gold cues. (1) is the basic configuration of our system with 3 labels and no dependencies, (2) adds dependency features to the model and (3) takes advantage of both dependency features and 2 additional labels.

We attribute the impact of dependency features to the finer-grained, syntactically close relation between events and the cues that negate them. While our n-gram features seem to overlap with the information provided by the graphs when it comes to consecutive chunks of tokens, we conclude that the relation between a negation cue and the word it focuses on is much more precisely modeled integrating the lexical patterns and lexical distance from the cue with contextual, dependency features.

7.4 Error analysis

7.4.1 Product Review

Errors in the PR system reflect the somewhat erratic nature of the annotations. The hardest challenge for our system is knowing where a scope ends; we find however that a very large portion of the errors produced by our system are often reasonable predictions of scope. This is because some of the examples in PR

directly contradict each other with regard to what "minimal scope" is. This is better explained by some examples (S-labeled tokens will hereon be displayed between pipes, e.g. "|token|"):

(7.10) [GOLD] If you are **not** yet a Dream Theater fan|,| [...]

(7.11) [SYS] If you are **not** yet |a| Dream Theater fan, [...]

(7.12) [GOLD] The beautiful Shizoko soon finds herself a willing participant in fulfilling **not** just |her| husband's fantasies but a slew of rich men.

(7.13) [SYS] The beautiful Shizoko soon finds herself a willing participant in fulfilling **not** just her husband's fantasies |but| a slew of rich men.

7.10 and 7.12 contain gold scope annotations, with 7.11 and 7.13 being their annotated counterparts. While *yet* and *just* occupy the same syntactic position, the human annotator makes a seemingly unfunded distinction in 7.10 and 7.12, including only the adverb in the former and the whole phrase it modifies in the latter.

While we have assessed that the finer grained label-set barely contributes to the system's overall performance, we observe that it makes quite a difference in a few scopes. Example 7.14 contains two correctly resolved scopes by our top-performing PR system; in 7.15 the same scopes are wrongly resolved by the system trained on the minimal label-set.

(7.14) That's just fine if you do**n't** like the post-black album cds |but| do**n't** put down the band |for| always trying new stuff.

(7.15) That's just fine if you do**n't** like the post-black album cds but do**n't** put down the band for always trying new stuff.

7.4.2 Conan Doyle

The most common error produced by the CD-trained model concerns discontinuous scopes. In Example 7.16 below, the coordinated second half of the sentence shares the same explicit subject "*I*", which the classifier fails to assign to the cue **not**; this kind of scope discontinuity has not been properly learned by the classifier, as we observe only one correct discontinuous scope assignment.

(7.16) I therefore spent the day at my club and did **not** return to Baker Street until evening|.|

This kind of error can probably be corrected, provided that the discontinuity is properly analyzed by the dependency parser, by identifying coordinations and retrieving missing dependencies traversing the graph during post-processing.

When it comes to negated events, we observe that errors are mostly due to either *predicting* an event for a *non-factual* context (false positive) or *not predicting* an event for a *factual* context (false negative), i.e., there are relatively few instances of predicting the wrong token for a factual context (which results in both a false negative and a false positive). This suggests that the CRF has learned what tokens should be labeled as an event for a negation, but has not learned so well how to determine whether the negation is factual or non-factual. In this respect it may be that incorporating information from a separate and dedicated component for factuality detection — as in the system of Read et al. (2012) — could yield improvements for the CRF event model.

7.5 Comparison with related work

To this point we have described experiments that aim at evaluating the isolated performance of our system, comparing only to a simplest-possible-system baseline and to a basic configuration of our system itself. It is however also interesting to measure its performance in comparison with other systems that address negation scope resolution using the same corpora, in order to assess whether the approach presented in this work is in fact a viable solution for resolving negation scope. Furthermore, since the results we compare with are for automatically resolved cues, it allows us to measure performance in a more realistic setting, without the advantage of gold cues.

7.5.1 Product Review

Comparing Product Review results to the scope resolution results reported by Councill, McDonald, and Velikovich (2010) directly might be unfair, since the real focus of their work is to evaluate the effects of negation scope resolution on sentiment analysis, rather than optimizing the scope resolution component itself. Their system is similar to ours, with a CRF labeler at its core and a feature-set comprised of lexical, distance-from-the-cue and dependency features. The label-set for their system has two symbols, one for in-scope and one for out-of-scope tokens. Our system expands on theirs in terms of more detailed syntactic, lexical and contextual features, and also with respect to negation modeling in terms of output labels.

In keeping with the results reported in Councill et al., we evaluate system outputs without converting the annotations into the CD format, using the same

PCS score described in Chapter 6.

Results for the PCS score reported by Councill et al. are shown in Table 7.7 Row (A), together with our own attempt at replicating their setup in Row (B). This is because the two systems have different pre-processing components and are not evaluated using the exact same script. In this respect the results in Row (B) are a special configuration of our system that uses only the O and N labels and does not take advantage of features other than the ones reported by Councill et al. for the CRF model. Both (B) and (C) systems have a lexicon-based negation cue classification component that takes advantage of the same lexicon described in Chapter 4. The results for Row (D) are for a system equal to (C) but with gold cues, providing an upper bound for comparisons.

	Configuration		**PCS**
(A)	C. et al., reported	class. cues	39.80
(B)	C. et al., reproduced	class. cues	42.37
(C)	Our System	class. cues	48.53
(D)	Our System	gold cues	67.85

Table 7.7: Comparative results on PR (A) is the PCS score reported by Councill, McDonald, and Velikovich (2010), (B) is a configuration of our system which replicates their settings and (C) is our best performing configuration. Negation cues are automatically resolved in (A), (B) and (C) using the lexicon presented in Chapter 4, while (D) is the same system configuration as (C) but with gold cues.

	P	**R**	**F_1**
Cues	71.19	100.0	83.17

Table 7.8: Lexicon-based cue detection results on PR.

Table 7.8 shows that the achille's heel of the very simple cue classifier is precision: while the perfect recall indicates that every instance of negation from the gold corpus is correctly retrieved, lexicon lookups generate many false-positives, which in this case are tokens that correspond to negation cues in the lexicon but do not generate any scopes. Secondly, comparing Rows (A) and (B) in Table 7.7 we measure the beneficial effect of the additional features and labels as slightly more than 6 percentage points.

7.5.2 Conan Doyle

For the Conan Doyle corpus, the setup for such a comparison is very straightforward: the output of two configurations of our system were submitted to *SEM shared task, ranking first in the open track and second in the closed track.

The system submitted to the closed track, where participants are not allowed to use resources outside of the ones that come with the corpus, uses syntactic representations obtained by converting the original constituency trees from CD into dependency graphs, while the output submitted to the open track uses Maltparser.

The reported scores for our system, UiO_2, in Table 7.9 are obtained on the held-out set in PR using cues retrieved by a Support Vector Machine-based classifier, which correctly retrieves 91.31% of the cues (Lapponi, Velldal, Øvrelid, and Read 2012). The CNS evaluation metric is the strictest one provided by the evaluation script, which counts perfect negation resolution on sentence level, i.e. the number of full sentences containing negation where cues, scopes, negated events are perfectly resolved.

System	%CNS
Closed Track	
UiO_1 r2	43.83
UiO_2	40.00
FBK	35.74
UWashington	34.04
UMichigan	27.23
UABCoRAL	26.81
Open Track	
UiO_2	41.28
UGroningen r2	27.23
UCM-1	18.72
UCM-2	11.91

Table 7.9: 2012 *SEM Shared Task rankings.

UiO_1, the top performing system, aligns scopes to constituents, using a machine-learning model to choose a syntactic unit among different candidates (Read, Velldal, Øvrelid, and Oepen 2012). This system addresses discontinuous scopes directly during post-processing with rules applied on the constituents. Additionally, UiO_1 has a dedicated factuality classifier, which is used to discard event-labeling in non-factual contexts. Similarly to our own system, FBK (Chowdhury 2012), UWashington (White 2012) and UMichigan (Abu-Jbara and Radev

2012) use a CRF classifier to resolve scopes and events, though we observe that our more involved approach in terms of feature and label selection does make a rather marked difference, with over 4 percentage point better CNS than FBK, the next best system.

Table 7.10 displays scope and event resolution results for our two systems in comparison with UiO_1 and FBK. Comparing UiO_1 event resolution results to our best performing system, we see the positive effects of their factuality classifier. Our system has higher precision, which means that it is better at not generating false positives, but is far worse in terms of recall, being nearly 18 percentage points behind UiO_1.

The scores for scope and event resolution are favorable to our UiO_2 system when comparing to FBK — while the differences are less pronounced than in the UiO_1 / UiO_2 comparison, the more generous CNS score difference indicates that our system is markedly better at assigning correct events and perfect scopes in full sentences.

System		**P**	**R**	**F_1**
UiO_1	Scopes	83.89	60.64	70.39
	Events	60.58	75.00	**67.02**
UiO_2, closed	Scopes	87.43	61.45	72.17
	Events	68.18	52.63	59.40
UiO_2, open	Scopes	85.71	62.65	**72.39**
	Events	66.90	57.40	61.79
FBK	Scopes	88.96	58.23	70.39
	Events	64.14	56.71	60.20

Table 7.10: *SEM Shared Task scope and event resolution results for our two submitted outputs, the top performing system and the one ranked below our own.

While our open and closed track systems are very similar for scopes alone, we see that the former has 2.39 percentage points advantage on the other latter for negated events. Looking at Table 7.11 we observe a more definite difference on CDD using gold cues, where our open track system achieves 80.55 F_1 score compared to the closed track's 74.88. Again, the only difference between these two systems is the source of the dependency graphs, converted C&J parses versus Malt parses.

A possible explanation for this difference is that the CRF benefits from having both the constituent and the dependency parses, as they are both recorded as

		Closed			Open		
		Prec	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**
(A)	Gold cues, CDD	81.72	69.09	74.88	83.65	77.68	**80.55**
(B)	Class. cues, CDE	68.18	52.63	59.40	66.90	57.40	61.79
(C)	(A), no const.	80.90	66.67	73.10	83.50	76.79	80.00

Table 7.11: Closed and open-track systems head-to-head event resolution on CDD with gold cues in (A) and on the held-out sets with classified cues in (B). (C) are the same configurations (A) without using the parse-tree constituents as features.

features in both configurations. While this is arguably redundant for the closed track model, where the dependency relation is just a conversion of the same parse, the open track model utilizes syntactic representation that are different in both nature and source. To evaluate this, we train closed and open track models without taking advantage of the constituent feature using the gold cues / CDD setup, showing negated events results in the third row of Table 7.11.

Contrarily to our intuition that the F_1 score difference between the two outputs would decrease by removing the constituent feature, we observe that it instead slightly increases from 5.67 to 6.9 percentage points. This might indicate that the better performance of the open track system is due to the quality of the dependency graphs; due to time constraints, we leave further evaluation for future work.

7.6 Summing it all up

We presented a series of scope resolution experiments on two corpora using significantly different annotation schemes. With gold cues, our top-performing system configuration yields an F_1 score of 80.22% for PR and 80.43% for CD, which suggests that our system scales well across different annotation paradigms. We evaluated the effects of adding features extracted using dependency graphs; while these features do not make much of a difference when resolving negation by PR standards, our results clearly show that they make a big difference for the more involved CD approach, with its concept of negated events as special tokens within the scope. This answers both Question 7.1 and 7.3; while perfor-

mance on scopes as simple sequences of tokens is comparable across domains and corpora, the more involved negation scopes of CD greatly benefit from the same dependency features that bring only marginal improvement to PR scope resolution.

The effect of the fine-grained label set was also evaluated and found to be beneficial across all our experiments, though not significantly so for PR.

Our system compares favorably to all known CRF-based based systems for negation scope resolution on PR and CD; the only machine learning-based system that achieves better overall results on CD utilizes a dedicated factuality classifier that efficiently disambiguates factual and non-factual contexts, resulting in improved negated event resolution.

Finally, we noted that the quality of the dependency representations appears to significantly affect the performance of negated event classification.

Chapter 8

Field Trip: Sentiment Analysis

Until now we have limited the scope of our research to building two parallel systems for negation scope resolution and evaluating the performance of different configurations. Like PoS tagging and syntactic parsing however, Negation Scope Resolution on its own is of limited interest, its real utility being that of augmenting the capabilities of other NLP systems.

As we have seen, the notion of negation scope is different in the two corpora that have been used in this work — maximal scopes in the Conan Doyle corpus and minimal scopes in Product Review. How do these two different takes on scope compare in a real-life task, and how can we take advantage of the notion of negated events introduced in the Conan Doyle corpus?

In this chapter we introduce one such task, present a simple system that attempts to tackle it, and evaluate the performance of the models presented in the previous chapter by using our scope resolution system as a sub-component.

8.1 Sentiment Analysis

Sentiment Analysis (SA) is an active field of research that focuses on the automatic detection and treatment of opinion and affection within NLP frameworks. It is important for a number of reasons. Improved search accuracy and recommendation systems can obviously benefit from it (users want to know how others feel about certain objects or services) and corporations are eager to collect intelligence on how their products are perceived. Furthermore, classifying information as "hateful" might help governments preventing crime, and grouping opinions as liberal or conservative allows for more accurate predictions on the outcomes of elections. Human Computer Interaction can benefit from it, as intelligent systems could boost their performance by adapting to the mood of the user.

The tasks that deal with sentiment, opinion and subjectivity show both similarities and contrasts with those of traditional information extraction and text categorization (Pang and Lee 2008). Say that we want to classify a document as being favorable or adverse towards some entity: the first challenge is to *extract* the parts of the text that bear the sentiment of the opinion holder towards said entity. Categorizing text as, for instance, belonging to one author rather than another can be seen as a parallel task to the one of classifying a review as either positive or negative, since the same machine learning techniques can be used to attack the problem.

One of the first and most influential experiments deals with document-level sentiment polarity classification (Turney 2002). This system uses a simple method based on Pointwise Mutual Information to determine the semantic orientation of phrases in, among other things, movie and car reviews: ”romantic ambience” being an example of positive semantic orientation and ”horrific events” one of the negative. The system achieves good results on the domain of cars yet barely surpasses the majority-class baseline for movies. Where the reviewer's satisfaction for a car appears to be a function of how satisfied she is with its parts, movies might have *evil characters* and *sad scenes* while still being masterpieces.

In general, Sentiment Analysis is a hard problem to tackle from a computational perspective because sentiment and opinion are subject to larger, more mutable contexts than, say, named entities and literary genres. The term *spandex* could for instance have been associated with positive sentiment in the eighties and used derogatorily in the nineties.

Sentiment analysis systems are also difficult to evaluate, as positive and negative sentiment, with all there is in between, is often in the eye of the beholder. An annotator who is asked to classify the polarity of a review into two classes is more likely to disagree with another than for other tasks; this results in lower annotator agreement scores (Bermingham and Smeaton 2009).

8.2 Negation and Sentiment Analysis

In its most obvious instance, the problem with negation and simple data-driven approaches in SA lies in polarity reversal: counting positive words such as *like* as indicators of positive sentiment, *I like this tablet* and *I do not like this tablet* would amount to the same results. This issue has been dealt with in different ways, for example by detecting the negation and creating a new *NOT_like* feature with inversed polarity (Pang, Lee, and Vaithyanathan 2002). Unfortunately, as we have seen, negation is much more complicated than that; determining its exact scope is essential to knowing what is being negated.

As mentioned in Chapter 4, Councill, McDonald, and Velikovich (2010) evaluate the effect of their negation resolution system on SA, matching n-grams to a lexicon of sentiment-bearing words and phrases in order to classify reviews as either positive or negative. The system is evaluated only on sentences that contain negation. Their baseline classifier is not negation-aware, achieving F_1 scores of 48.8 and 32.3 on positive and negative sentiment respectively. The negation-aware one, which flips the polarity of lexicon-matches when they are found to be in scope, improves the score for positive sentiment classification by 29.5 percent and 11.4 percent for negative sentiment classification.

This experiment shows the efficacy of supervised machine-learning systems for Negation Scope Resolution as sub-components to Sentiment Analysis. The scope-resolution behavior of such systems, however, depends on the nature of the annotations they have been trained on. What we are interested in evaluating here is how the different conceptions of negation from the Product Review and Conan Doyle corpora translate into performance in a Sentiment Analysis Task.

8.3 A simple system for document-level sentiment polarity classification

To evaluate the performance of our scope resolution system on Sentiment Analysis, we build a simple system based on the Polarity Dataset v.2.0 (Pang, Lee, and Vaithyanathan 2002), a collection of 1000 positive and 1000 negative reviews. This corpus is run through our best performing PR and CD-based systems in order to obtain negation scopes; negation cues are classified using the lexicon described in Chapter 4.[1]

Sentiment classification is done via a scoring function that takes advantage of AFINN-111 (Nielsen 2011), a lexicon of 2477 manually annotated sentiment-bearing words. Each entry in the lexicon is annotated with an integer between -5 five and 5, according to the positive or negative degree of its inherent sentiment, Table 8.1 shows an excerpt from the lexicon with scores ranging from -5 to 5.

We match each word in a review to the lexicon and collect the returned integer; if the word is found to be in a negation scope, we simply multiply the score with -1, inverting its polarity. A review is deemed positive if the sum of this token-wise score is greater than 0, negative if it is less than 0.

After some experimentation with this simple approach, we found that factoring the count of positive and negative lookups in the scoring function yields better results; with $s(\mathrm{w})$ returning the lexicon-match score and $c(\mathrm{pos})$ and $c(\mathrm{neg})$

[1] To achieve this, each review file in the Polarity Dataset has been converted to same format of the Conan Doyle Corpus, which should facilitate further experimentation.

Word	Score
bastard	-5
fraudulent	-4
greed	-3
helpless	-2
mandatory	-1
motivate	1
opportunity	2
pleasant	3
rejoice	4
superb	5

Table 8.1: an excerpt from the AFINN-111 lexicon with scores ranging from -5 to 5.

the number of lookups returning a positive or a negative integer respectively, the global score gb of a review R is calculated as follows:

$$gb(\mathrm{R}) = c(\mathrm{pos}) - c(\mathrm{neg}) + \sum_{w \in R} s(\mathrm{w}) \tag{8.1}$$

While such a simple approach can hardly be considered state-of-the-art in the Sentiment Analysis domain, it does allow us to simply and effectively evaluate how our negation-scope resolution system performs as a polarity-inverter, and how the data-set it is trained on affects the results.

	It	ca	n't	fall	under	drama	,	comedy	,	thriller	or	action
Lex. score	-	-	-	-	-	-	-	1	-	-	-	-
PR-scopes	O	O	CUE	N	N	N	S	O	O	O	O	O
CD-scopes	N	N	CUE	N	N	N	N	N	N	N	N	N
CD-events	O	O	CUE	E	O	O	O	O	O	O	O	O

Figure 8.1: A sentence from the Polarity Dataset with scores from the lexicon and the negation annotations from the three different models.

Figure 8.1 shows one sentence from the Polarity Dataset with scores from the lexicon (in this particular instance, only matches one word) and the negation annotations from the three different models.

8.4 Experiment

Four configurations of the system described in the previous section were evaluated with polarity-wise F_1 score, using the non negation-aware configuration as a baseline. False positives are counted when a review is classified with the wrong polarity; rather than assigning reviews where the score from the lexicon sums to zero to one of the two classes, we count such outcomes as false negatives. One configuration utilizes negation scopes obtained from our PR-based model for polarity reversal, while the other two are based on the CD-model and use scopes and negated event respectively.

	positive	**negative**
total matches	43582	40527
PR-scopes inversion	3065	1000
CD-scopes inversion	4274	4822
CD-events inversion	789	897

Table 8.2: Token-wise counts of AFINN-111 lexicon matches on the Polarity Dataset. The first row shows the total amount, while the last three show the number of polarity inversions in the three negation-aware configurations.

Table 8.2 shows the total amount of matches from the lexicon for both positive and negative reviews and how many times their score is inverted in each negation-aware configuration. In this respect, they all perform as expected, with *CD-scopes* being the most inversion-eager configuration, *CD-events* the most conservative and *PR-scopes* something in between the two.

Table 8.3 shows results for all four configurations. At a glance, we see that the system is biased towards positive polarity, with F_1 score for positive reviews being consistently higher than it is for the negative ones. All configurations that incorporate polarity inversion outperform the baseline for negative reviews, while none outperforms the positive baseline. CD-scopes obtains the best overall performance, improving on the negative baseline with almost 10 percentage points.

This results are noteworthy because the PR system is trained on essentially the same kind of text found in the reviews we are classifying, which intuitively should make it more apt to aid in this domain. Besides, although the Product Review corpus is in part conceived with Sentiment Analysis in mind, the maximal, syntactically motivated scopes from CD appear to work better as polarity inverters.

config.		**P**	**R**	**F$_1$**	cmb **F$_1$**
Baseline	pos	79.65	99.12	88.33	75.95
	neg	47.11	97.69	63.57	
PR-scopes	pos	78.00	98.84	87.19	78.62
	neg	54.77	97.11	70.04	
CD-scopes	pos	77.38	98.19	86.55	79.29
	neg	57.04	97.74	72.04	
CD-events	pos	79.25	99.11	88.08	77.32
	neg	50.35	98.22	66.57	

Table 8.3: Results of the document-level sentiment polarity classifier. While no negation-aware configuration outperforms the baseline for the positive reviews, the margin of improvement on negative review classification is large enough that it translates in consistent overall improvements over the baseline. The configuration that inverts the polarity of most words, CD-scopes, yields the best overall results.

The same CD configuration, however, is also the weakest on positive reviews, with almost 2 percentage points lower F_1 than the baseline; the converse is true for the CD-events configuration. Here, the F_1 gain for negative reviews is measured in 3 percentage points, while the loss for positive reviews is less than 0.2 points. The performance of the PR-scopes configuration lies somewhere in between the other two in all respects.

The outputs of the classifier were tested for significance with $p = 0.05$ using the same sign test described in Chapter 6; both PR-scopes and CD-scopes are significantly different from the baseline and the CD-events configurations; the opposite is true for the [PR-scopes, CD-scopes] and [baseline, CD-events] pairs.

Although all negation-aware configurations outperform the baseline in terms of the combined positive/negative F_1 score, it is interesting and somewhat surprising that no configuration yields any improvement on positive review classification, while the benefits for negative reviews seem to be proportional to the breadth of the negation scopes generated by the different models. This could be related with to the way negation is used in positive and negative contexts, or simply to the fact that our naive approach does not harness the benefits of Negation Scope Resolution fully. Finer-grained sentiment analysis experiments

and thorough error analysis could shed some light in these respects; due to time constraints, however, we leave further investigation for future work.

Chapter 9

Conclusion

In this work we presented a machine learning-based system for negation scope resolution that automatically determines which parts of a given sentence are affected by the presence of an explicit negation cue. The problem was formulated as a sequence labeling one; motivation for this approach was given in the form of an introduction to sequence labeling techniques for Natural Language Processing in Chapter 2. Our models for scope resolution rely heavily on dependency graphs as a source of syntactic information; this syntactic structure was presented in Chapter 3.

Our approach to negation scope resolution was evaluated on corpora that differ in both domain and, most importantly, the conception of what negation scope is; detailed information on this corpora and their differences was provided in Chapter 4. We provided an in-depth system description in Chapter 5, explaining every step of the data processing, illustrating and motivating our choices in terms of feature engineering and internal representation.

Our system was evaluated on several levels in Chapters 7 and 8; all the evaluation metrics used in this work were introduced in Chapter 6. First, we evaluated several configurations against a simple baseline and a minimal configuration of the system itself, utilizing gold negation cues in order to isolate the scope-resolution performance. Then, we compared its performance to similar systems using different cue-resolution strategies. Finally, we assessed the efficacy of different configurations as subcomponents in another NLP task.

Experimenting with different configurations of our system, we gathered the following insights:

Our sequence-labeling approach scales well across the different scope annotation paradigms.

Negation scopes in PR are consecutive chunks of tokens, which in the vast majority of cases start with a negation cue; in CD on the other hand, scopes can propagate to either side of a cue and contain special tokens, called negated events, in factual contexts. While sequence labeling certainly is an intuitive choice to automatically annotate CD-scopes, our experiments show that our system can be applied to both corpora with results that are slightly above the 80% F_1 score mark in both cases, when taking advantage of gold cues.

The effects of syntactic features and additional labels vary across corpora.

The experiments conducted in this work show that the syntactic features and additional labels significantly improve the performance of the sequence labeler for CD-scopes, greatly so when it comes to negated event classification. This appears not to be the case for the simpler CD-scopes, where the very small observed improvements were not deemed to be statistically significant.

The source of syntactic analyses noticeably affects the performance of negated event classification.

We experimented with two different sources of dependency parses on CD, with dependency analyses coming from Maltparser resulting in better event resolution compared to those obtained by converting parse trees from the Charniak and Johnson (2005) parser into dependency graphs. While the reasons behind this difference in performance are yet to be assessed, this is an interesting result and suggests that machine learning-based scope resolution might be a viable option for extrinsic parser evaluation.

As a polarity inverter in a simple Sentiment Analysis system, The model obtained from CD performs better than the one obtained from PR.

We have built a simple document-level sentiment-polarity classifier, detailed in Chapter 8, in order to evaluate the performance of the models obtained from PR and CD as sub-components in another task. While the text we automatically annotate with negation scopes is from the same domain as the one in PR, we obtain the best results inverting the polarity of sentiment-bearing words using the

CD-model. This is noteworthy considering that PR, with its minimal approach negation scope, was developed in the context of sentiment analysis.

Our system compares favorably to other similar, sequence labeling-based approaches.

We compared our results to the ones reported by Councill, McDonald, and Velikovich (2010) on PR by replicating their settings and taking advantage of the same lexicon-based cue classification component. Our system, which extends on theirs both in terms of lexical and syntactic features and label-set granularity, performs better in our experiments, indicating that more detailed feature engineering can make a noticeable difference with negation scope resolution.

In the 2012 *SEM shared task on negation resolution, which used CD for evaluation, our system was ranked first in the open track and second in the closed track, outperforming all other CRF-based systems. The best overall submission, UiO_1 (Read et al. 2012), which approaches scope resolution from a completely different angle, directly addresses the factual nature of negated events, resulting in vastly superior event resolution. In terms of scopes alone, however, their system does not outperform ours.

The good overall performance of our system makes it both an adequate tool for further experiments and, we hope, a valid contribution to the ongoing research on negation within the Natural Language Processing community.

9.1 Improving the system

There are several sides of our system that we plan on developing further. We want to extend our feature set with more contextual features, exploiting phrase-structure trees in addition to the dependency graphs to model the relation between cues and tokens. Moreover, there are other negation-specific contextual features that could be exploited, for instance by looking at negative polarity items like *any* or *at all* as a source of information. Encouraged by the results obtained on CD, we also plan on further developing the label set, and continue experimenting with grouping tokens as a complement to feature extraction.

We would like to improve event detection by taking advantage of the factuality classifier used by UiO_1 and, like in their system, directly address discontinuous scopes. Another possibility is to incorporate their features into the CRF model.

We also plan on evaluating the performance of the system as a subcomponent in NLP tasks other than Sentiment Analysis, for example textual entailment or

information extraction.

9.2 Future Work

The avenues for future work outside of improving this system are also plentiful. Since they can be a useful contribution to negation-related research, we plan to make our gold-scope annotated, pre-processed version of PR available, together with the negation-scope annotated versions of the Polarity Dataset. In the same spirit, we are working with refactoring and optimizing the source code of our system so that it can be made available to the community as open-source software.

Furthermore, the observed differences in performance across different dependency sources demand more investigation. Thoroughly evaluating the system using several off-the-shelf dependency parsers could shed some light in this respect, while at the same time provide insights on the extrinsic performance of the parsers themselves.

Finally, we would like to continue to work with negation in the NLP domain by concentrating on the real-life implementation and optimization of negation resolution components within existing frameworks, widening the breadth of our research by focusing on languages other than English.

Bibliography

Abu-Jbara, A. and D. Radev (2012). UMichigan: A Conditional Random Field Model for Resolving the Scope of Negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, Montreal, pp. 328–334.

Bermingham, A. and A. F. Smeaton (2009). A study of inter-annotator agreement for opinion retrieval. In J. Allan, J. A. Aslam, M. Sanderson, C. Zhai, and J. Zobel (Eds.), *SIGIR*, pp. 784–785. ACM.

Bird, S., E. Klein, and E. Loper (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Beijing: O'Reilly.

Charniak, E. and M. Johnson (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the Forty-Third Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI.

Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.

Chowdhury, M. F. M. (2012). FBK: Exploiting Phrasal and Contextual Clues for Negation Scope Detection. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, Montreal, pp. 340–346.

Councill, I. G., R. McDonald, and L. Velikovich (2010). What's great and what's not: Learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop On Negation and Speculation in Natural Language Processing*, Uppsala, pp. 51–59.

de Marneffe, M.-C. and C. D. Manning (2008a). Stanford dependencies manual. Technical report.

de Marneffe, M.-C. and C. D. Manning (2008b). The stanford typed dependencies representation. In *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271.

Dridan, R. and S. Oepen (2012). Tokenization: Returning to a long solved problem — a survey, contrastive experiment, recommendations, and toolkit. Jeju, Republic of Korea, pp. 378–382.

Gildea, D. and D. Jurafsky (2002). Automatic labeling of semantic roles. *Computational Linguistics 28*(3), 245–288.

Horn, L. R. (2010). *The Expression of Negation*. Walter de Gruyter.

Horn, L. R. and Yasuhiko (2000). *Negation and Polarity - Syntactic and Semantic Perspectives*. Oxford University Press.

Jurafsky, D. and J. Martin (2000). Speech and language processing: An introduction to natural language processing, computational linguistics and speech. NJ, USA: Pearson Prentice Hall.

Kaplan, R. M. (2005). A method for tokenizing text. *Inquiries into words, constraints and contexts 1*, 55–64.

Lapponi, E., E. Velldal, L. Øvrelid, and J. Read (2012). UiO$_2$: Sequence labeling negation using dependency features. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, Montreal, pp. 319–327.

Lavergne, T., O. Cappé, and F. Yvon (2010). Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 504–513. Association for Computational Linguistics.

Manning, C. D. and H. Schütze (1999). *Foundations of statistical natural language processing*. Cambridge, MA, USA: MIT Press.

Marcus, M., M. Marcinkiewicz, and B. Santorini (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 19*(2), 313–330.

Morante, R. and E. Blanco (2012). *SEM 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, Montreal.

Morante, R. and W. Daelemans (2009). A metalearning approach to processing the scope of negation. In *CoNLL '09: Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pp. 21–29.

Morante, R., S. Schrauwen, and W. Daelemans (2011). Annotation of negation cues and their scope. guidelines v1.0.

Morante, R. and C. Sporleder (2012). Modality and negation: An introduction to the special issue. *Computational Linguistics 38*(2), 1–38.

Nielsen, F. Å. (2011). A new anew: Evaluation of a word list for sentiment analysis in microblogs. *CoRR abs/1103.2903*.

Nivre, J. (2005). Dependency grammar and dependency parsing. Technical report, Växjö University: School of Mathematics and Systems Engineering.

Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryiğit, S. Kübler, S. Marinov, and E. Marsi (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering 13*(2), 95–135.

Pang, B. and L. Lee (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval 2*(1-2), 1–135.

Pang, B., L. Lee, and S. Vaithyanathan (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, pp. 79–86.

Read, J., E. Velldal, L. Øvrelid, and S. Oepen (2012). UiO_1: Constituent-based discriminative ranking for negation resolution. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, Montreal, pp. 310–318.

Skjærholt, A. (2011). Ars flectandi: Automated morphological analysis of latin. *Master's thesis, University of Oslo*.

Stone, P. J. (1966). *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

Sutton, C. and A. McCallum (2010). An introduction to conditional random fields. *Arxiv preprint arXiv:1011.4088*.

Tottie, G. (1991). *Negation in English Speech and Writing: A Study in Variation (Quantitative Analyses of Linguistic Structure)*. Academic Press.

Tsuruoka, Y. (2005). Bidirectional inference with the easiest-first strategy for tagging sequence data. In *In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 467–474.

Turney, P. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the Association for Computational Linguistics (ACL)*, Philadelphia, pp. 417–424.

Velldal, E., L. Øvrelid, J. Read, and S. Oepen (2012). Speculation and negation: Rules, rankers, and the role of syntax. *Computational linguistics 38*(2), 369–410.

Vincze, V., G. Szarvas, R. Farkas, G. Móra, and J. Csirik (2008). The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics 9*(S-11).

White, J. P. (2012). UWashington: Negation Resolution using Machine Learning Methods. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, Montreal, pp. 335–339.

Zwicky, A. M. (1985). Heads. *Journal of Linguistics 21*, 1–29.